# Editoria elettronica con AlmI

**Dal foglio di carta alla realizzazione di un documento secondo AlmI, con le note necessarie per scrivere in modo coerente rispetto all'opera «Appunti di informatica libera»**

Daniele Giacomini *daniele @ swlibero.org*

2003.06.29

**Appunti Linux**



**Appunti di informatica libera**



**Editoria elettronica con Alml**



Via Morganella Est, 21 -- I-31050 Ponzano Veneto (TV) -- *daniele @ swlibero.org*

The main distribution for *Appunti di informatica libera* is described below. For every distribution channel the maintainer's name and address is also reported.

### Ordering

Prosa srl distribute *Appunti di informatica libera* and possibly oder related products through the commercial channel. Any request to buy something related to *Appunti di informatica libera*, can be addressed to Prosa srl:

<*http://www.prosa.it/*>

<*http://www.opencdpress.com/store/prosa*>

### Internet mirrors with good bandwidth

- direct reading: <*http://www.lingueperilweb.com/AppuntiLinux/HTML/a2.html*>
  download: <*http://www.lingueperilweb.com/AppuntiLinux/*>
  Fabrizio Rubino, `fabrizio @ koalasoft.org`
- direct reading: <*http://rm.mirror.garr.it/mirrors/AppuntiLinux/HTML/a2.html*>
  download: <*http://rm.mirror.garr.it/mirrors/AppuntiLinux/*>
  download: <*ftp://rm.mirror.garr.it/mirrors/AppuntiLinux/*>
  `mirror-service @ garr.it`
- direct reading: <*http://www.informaticalibera.it/a2/HTML/a2.html*>
  download: <*http://www.informaticalibera.it/a2/*>
  Fabrizio Rubino, `fabrizio @ koalasoft.org`
- direct reading: <*http://appunti.linux.it/*>
  download: <*http://ftp.linux.it/pub/mirrors/appunti/*>
  Italian Linux Society, `info @ linux.it`

### Internet main distribution sites

- direct reading: <*http://a2.swlibero.org/*>
  download: <*ftp://a2.swlibero.org/a2/*> and <*http://a2.swlibero.org/ftp/*>
  Michele Dalla Silvestra, `mds @ swlibero.org`
- direct reading: <*http://www.koalasoft.org/AppuntiLinux/HTML/*> and <*http://lnx.koalasoft.org/AppuntiLinux/HTML/*>
  download: <*http://www.koalasoft.org/AppuntiLinux/*> and <*http://lnx.koalasoft.org/AppuntiLinux/*>
  Fabrizio Rubino, `fabrizio @ koalasoft.org`
- direct reading: <*http://appuntilinux.torino.linux.it/*>
  download: <*ftp://ftp.torino.linux.it/appunti-linux/*>
  Carlo Perassi, `carlo @ linux.it`
- direct reading: <*http://www.a2.prosa.it/*>
  download: <*ftp://ftp.a2.prosa.it/a2/*>
  Davide Barbieri, `paci @ prosa.it`
- direct reading: <*http://sansone.crema.unimi.it/linux/a2/HTML/*>
  download: <*http://sansone.crema.unimi.it/linux/a2/*>
  Fabrizio Zeno Cornelli, `zeno @ filibusta.crema.unimi.it`

- direct reading: <*http://www.pctime.it/servizi/appunti-linux/*>
  download: <*http://www.pctime.it/servizi/appunti-linux/a2-prelievo/*>
  Franco Lazzero, PCTIME, *pctime @ pctime.net*
- direct reading: <*http://linux.pueste.it/*>
  download: <*http://linux.pueste.it/filearea/AppuntiLinux/*>
  David Pisa, *david @ iglu.cc.uniud.it*
- direct reading: <*http://www.informasiti.com/Appunti/HTML/*>
  download: <*http://www.informasiti.com/Appunti/*>
  Claudio Neri, Sincro Consulting, *neri.c @ sincroconsulting.com*

**GNU distributions**

- GNU/Linux **Debian** <*http://packages.debian.org/appunti-informatica-libera*>
  Massimo Dal Zotto, *dz @ cs.unitn.it*

**Italian magazine's CD-ROM**

- **inter-punto-net** <*http://www.interpuntonet.it*>
  Michele Dalla Silvestra, *mds @ swlibero.org*
- **Internet News** <*http://inews.tecnet.it*>
  Francesco Facconi, *francescofacconi @ libero.it*
  Fabio Ferrazzo, *fabio.fr @ tiscalinet.it*
- **Linux Magazine** <*http://www.edmaster.it/index.php?job=prodotti&id=5*>
  Emmanuele Somma, *esomma @ ieee.org*
- **Linux Pro** <*http://www.linuxpro.it/*>
  Massimiliano Zagaglia, *max @ linuxwaves.com*
- **PC Upgrade** <*http://www.pcupgrade.it/*>
  *info @ pcupgrade.it*

# Indice generale

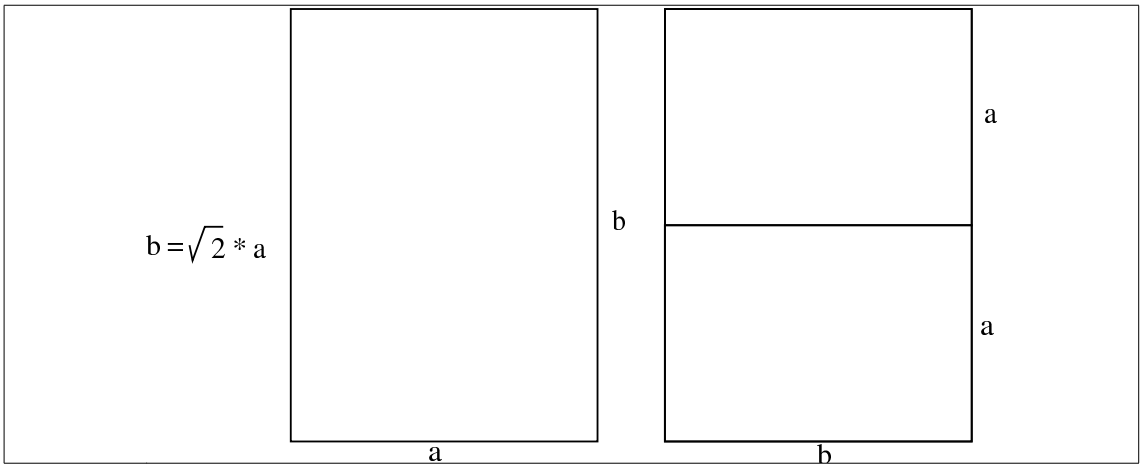
# Formati standard della carta

Lo standard ISO 216, così come gli standard UNI 936 e DIN 476, definisce i formati di carta più comuni, secondo una logica molto semplice. Vale la pena di riassumere brevemente i concetti legati a questo standard, ancor prima di affrontare altri problemi legati alla scrittura.

## 1.1 Caratteristiche fondamentali dello standard ISO 216

Nello standard ISO 216, i lati del foglio di carta hanno un rapporto fisso, dove il lato lungo è pari alla radice quadrata di due (circa 1,4142) per la lunghezza del lato corto (figura 1.1).

Figura 1.1. Il rapporto tra i lati di un foglio ISO 216.




Questo rapporto ha una proprietà importante, che consente al foglio di carta di essere dimezzato sul lato lungo, oppure di essere raddoppiato sul lato corto, mantenendo lo stesso rapporto tra i lati.

Lo standard ISO 216 definisce tre diverse serie di questi formati, ognuna delle quali parte da una dimensione di partenza, generando le altre dimensioni suddividendo quella precedente a metà, sul lato lungo. La serie A, ha come punto di riferimento il formato A0, corrispondente a un foglio con un'area di un metro quadro, tuttavia non si tratta del formato più grande, che è ottenuto raddoppiando due volte il formato A0, ottenendo così quattro metri quadri.

La tabella 1.1 elenca le dimensioni di tutti i formati delle tre serie, denominate A, B e C. Come si può osservare, i valori sono approssimati al millimetro, in aderenza al SI (sezione 3.5).

Tabella 1.1. ISO 216: formato A, B e C.

| A | mm | B | mm | C | mm |
|---|---|---|---|---|---|
| 4A0 | 1682 x 2378 | -- | -- | -- | -- |
| 2A0 | 1189 x 1682 | -- | -- | -- | -- |
| A0 | 841 x 1189 | B0 | 1000 x 1414 | C0 | 917 x 1297 |
| A1 | 594 x 841 | B1 | 707 x 1000 | C1 | 648 x 917 |
| A2 | 420 x 594 | B2 | 500 x 707 | C2 | 458 x 648 |
| A3 | 297 x 420 | B3 | 353 x 500 | C3 | 324 x 458 |
| A4 | 210 x 297 | B4 | 250 x 353 | C4 | 229 x 324 |
| A5 | 148 x 210 | B5 | 176 x 250 | C5 | 162 x 229 |
| A6 | 105 x 148 | B6 | 125 x 176 | C6 | 114 x 162 |
| A7 | 74 x 105 | B7 | 88 x 125 | C7 | 81 x 114 |
| A8 | 52 x 74 | B8 | 62 x 88 | C8 | 57 x 81 |

| A | mm | B | mm | C | mm |
|---|---|---|---|---|---|
| A9 | 37 x 52 | B9 | 44 x 62 | C9 | 40 x 57 |
| A10 | 26 x 37 | B10 | 31 x 44 | C10 | 28 x 40 |

## 1.2 Utilizzo pratico dei vari formati ISO 216

Tabella 1.2. Esempi di utilizzo pratico dei vari formati.

| Formati | Utilizzo |
|---|---|
| A0, A1 | Disegno tecnico; poster. |
| A2, A3 | Disegno; diagrammi; tabelle di grandi dimensioni. |
| A4 | Lettere; riviste; cataloghi; carta per stampanti comuni e per fotocopiatrici. |
| A5 | Blocchi per appunti. |
| C4 | Buste per il formato A4. |
| C5 | Buste per il formato A4 piegato a metà. |
| C6 | Buste per il formato A4 piegato due volte. |
| B4, A3 | Giornali. |

La percentuale di ingrandimento o di riduzione di un formato per ottenerne un altro, si determina facilmente, tenendo conto che si sta facendo riferimento all'ampiezza e all'altezza del foglio, non alla sua area. In pratica, riducendo un formato A4 al 50 %, si ottiene un formato A6, mentre per arrivare al formato A5 occorre usare una riduzione al 71 %. In altri termini, 71 %, ovvero 0,71, approssima la radice quadrata di 0,5. La tabella 1.3 riepiloga alcune trasformazioni tipiche, da un formato a un altro dello standard ISO 216.

Tabella 1.3. Esempi di ingrandimento e riduzione dei formati più comuni.

| Trasformazione richiesta | rapporto | percentuale (approssimata) |
|---|---|---|
| da A***n*** a A***n***+1 | sqrt(0,5) | 71 % |
| da B***n*** a A***n*** | sqrt(sqrt(0,5)) | 84 % |
| da A***n*** a B***n*** | sqrt(sqrt(2)) | 119 % |
| da B***n*** a A***n*** - 1 | sqrt(sqrt(2)) | 119 % |
| da A***n*** a A***n*** - 1 | sqrt(2) | 141 % |

La massa di un foglio di serie A, può essere determinata facilmente, sapendo che A0 ha una superficie di un metro quadro. In pratica, basta conoscere la densità superficiale della carta (la cosiddetta grammatura) che si esprime normalmente in grammi per metro quadro, dividendone opportunamente il valore: l'A***n*** avrà una massa pari a $2^{-n}$ volte quella dell'A0. Per esempio, la massa di un foglio A4 sarà $2^{-4}$ volte quella di un A0; ovvero 1/16; se la grammatura è 80 g/m$^2$, la massa di un foglio A4 è 5 g.

Le dimensioni dei fogli delle tre serie ISO 216 possono essere determinate anche attraverso delle formule matematiche, come mostrato nella tabella 1.4. Si osservi che le misure che si ottengono sono espresse in metri.

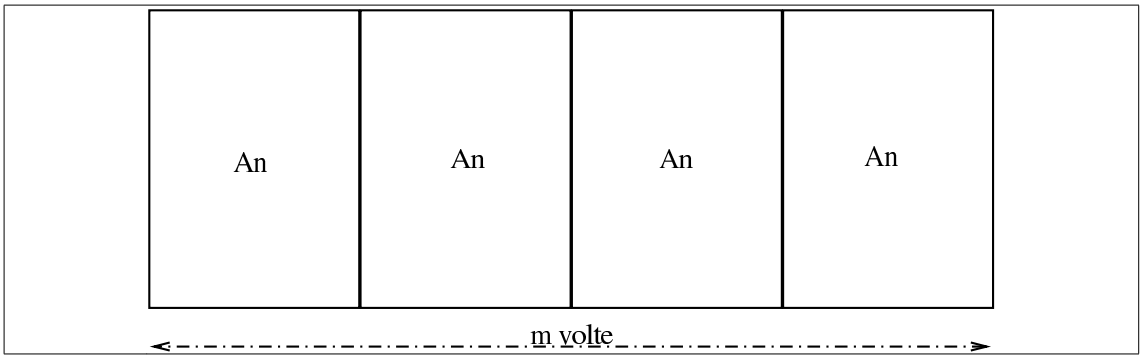
Tabella 1.4. Formule per calcolare le dimensioni della carta secondo lo standard ISO 216.

| Formato | Ampiezza in metri | Altezza in metri |
|---|---|---|
| A***n*** | $2^{(-1/4-n/2)}$ | $2^{(1/4-n/2)}$ |
| B***n*** | $2^{(-n/2)}$ | $2^{(1/2-n/2)}$ |
| C***n*** | $2^{(-1/8-n/2)}$ | $2^{(3/8-n/2)}$ |

## 1.3 Formati multipli

Quando non si può utilizzare un formato in cui il rapporto tra la lunghezza dei lati sia quello delle serie A, B o C comuni, si possono usare dei multipli di uno di questi formati. Come si vede nella figura 1.2, si tratta di affiancare più fogli di un certo formato, estendendo il lato corto. Questi formati estesi si indicano come A***n***x***m***, dove ***m*** rappresenta quanti fogli di tipo A***n*** affiancare. Per esempio, il formato A3 è equivalente al formato A4x2.

Figura 1.2. Formati multipli A***n***x***m***.




## 1.4 Riferimenti

- Markus Kuhn, *International standard paper sizes*
  *<http://www.cl.cam.ac.uk/~mgk25/iso-paper.html>*
- *Guide to international paper sizes, Concise tables of measurements*, EDS Inc., 1997-2000
  *<http://www.twics.com/~eds/paper/papersize.html>*
- R. Smith, F. Wright, T. Hastings, S. Zilles, J. Gyllenskog, *RFC 1759: Printer MIB, Appendix B - Media size names from ISO/IEC 10175 Document printing architecture*, 1995
  *<http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1759.html>*
  *<http://www.cis.ohio-state.edu/cs/Services/rfc/rfc-text/rfc1759.txt>*
- T. Hastings, R. Herriot, R. deBry, S. Isaacson, P. Powell, *RFC 2911: Internet printing protocol/1.1: model and semantics, Appendix C: "media" keyword values*, 2000
  *<http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2911.html>*
  *<http://www.cis.ohio-state.edu/cs/Services/rfc/rfc-text/rfc2911.txt>*

# Nozioni elementari di tipografia

Prima di studiare un programma di editoria elettronica conviene conoscere almeno qualche nozione di tipografia. Studiando la natura del problema si può comprendere la ragione di alcuni comportamenti dei programmi più raffinati che rispecchiano nella loro impostazione la filosofia della tipografia tradizionale.

## 2.1 Caratteri

Il ***carattere*** è qualunque segno grafico utilizzato in tipografia per rappresentare le lettere, i segni di interpunzione, le cifre e altri grafemi. La conoscenza delle caratteristiche fondamentali del carattere da stampa è necessaria per poter comprendere il funzionamento e la logica dei programmi di composizione tipografica. Sul carattere si possono distinguere diversi aspetti, in particolare:

- specie alfabetica;
- stile, o gruppo stilistico;
- serie alfabetica, o variante di serie;
- scala dimensionale.

Al di sopra di questa classificazione sta eventualmente il ***genere***, intendendo con questo la distinzione in base ai suoi componenti: segni alfabetici, segni paralfabetici, segni estralfabetici, fregi, iconografie, paraiconografie.

### 2.1.1 Specie alfabetica

La ***specie*** è una collezione di segni di un tipo di scrittura. Per quanto ci riguarda, la specie alfabetica comune è quella dell'alfabeto latino. All'interno di una specie alfabetica si possono distinguere diverse collezioni alfabetiche, per esempio come nella distinzione tra lettere maiuscole e minuscole che avviene nell'alfabeto latino.

Dalla differenza tra gli alfabeti nasce a volte la necessità di rendere un testo attraverso un alfabeto alternativo. La ***traslitterazione*** è il procedimento di traslazione da un sistema alfabetico a un altro, in modo da ricomporre un testo facendo uso di un sistema alfabetico diverso da quello originale. La traslitterazione punta a riprodurre un testo in modo che sia possibile in qualsiasi momento il procedimento inverso per riottenere il testo originale. Il caso più comune in cui si ha la necessità di utilizzare la traslitterazione è quello della citazione in cui l'originale utilizza un alfabeto esotico per il quale non si dispone del carattere tipografico. Come si può immaginare, la traslitterazione è regolata da norme internazionali.

### 2.1.2 Gruppo stilistico

Una volta definita la specie di un carattere si possono distinguere delle varianti che riguardano lo ***stile***, ovvero il disegno e il suo gusto estetico. Sull'alfabeto latino sono stati realizzati una quantità così grande di stili diversi che è difficile persino riuscire a classificarli. In generale vi si fa riferimento attraverso il nome. Gli stili più noti nella composizione elettronica sono: Times, Helvetica e Courier.

I tre nomi citati rappresentano oggi, simbolicamente, le caratteristiche fondamentali di uno stile: la presenza o l'assenza di grazie e la proporzionalità o meno della larghezza dei segni.[1]

Le grazie sono dei piedini terminali che hanno lo scopo di abbellire il carattere e di guidare la vista durante la lettura. Il Times è il tipico stile con grazie, mentre Helvetica è il suo opposto.

I segni dei caratteri da stampa sono generalmente di larghezza diversa; solo le prime forme di scrittura meccanica, come la macchina da scrivere e le prime stampanti, hanno creato la necessità di utilizzare dei simboli a larghezza uniforme. Il Courier è il rappresentante di questo tipo di stile a larghezza fissa.

In generale, uno stile riguarda esclusivamente una specie alfabetica, ma quando uno stile assume importanza e notorietà, può succedere che venga adottato anche da altre specie. Per questo si può distinguere tra Times Roman (Times New Roman), Times Greco, Times Cirillico e altri. Il primo tra quelli citati è ovviamente il Times dell'alfabeto latino.

### 2.1.3 Serie

La ***serie alfabetica***, o la variante di serie, rappresenta una distinzione all'interno di uno stile, in base alla ***forma***. Le forme comuni di uno stesso stile riguardano la pendenza, il tono e la larghezza.

- La pendenza si riferisce all'inclinazione delle aste e si distingue generalmente tra ***tondo***, che rappresenta un carattere con aste verticali, e ***corsivo*** in cui le aste sono inclinate in avanti. Generalmente, l'aspetto dei caratteri di un corsivo, pur restando all'interno della stesso stile, è abbastanza diverso da quello del tondo. Quando si utilizza un sistema di composizione elettronico può capitare di avere a disposizione uno stile nel quale manchi il corsivo, che però viene ottenuto in qualche modo distorcendo il tondo. In questo caso si parla preferibilmente di carattere «inclinato» in modo volutamente generico.
- Il tono, o lo spessore, rappresenta l'intensità del carattere che si percepisce visivamente. Essendo un concetto che deriva dalla stampa con inchiostro nero, si distingue generalmente tra ***chiarissimo***, ***chiaro***, ***nero*** (***neretto***) e ***nerissimo***.
- La larghezza è una caratteristica di cui dispongono solo alcuni stili, ovvero li può riguardare direttamente, nel senso che uno stile per sua natura può essere «stretto» o «largo». In base alla larghezza si distinguono solitamente: lo ***strettissimo***, lo ***stretto***, il normale, il ***largo*** e il ***larghissimo***.

È bene chiarire che ogni stile può disporre o meno di varianti seriali adatte. Alcuni stili, spesso riferiti a specie alfabetiche simboliche, dispongono di una serie unica.[2]

[1] Questi tre stili sono molto importanti, in parte per motivi storici, ma soprattutto perché sono quelli che si hanno a disposizione più di frequente.

[2] Alcuni sistemi di composizione riescono a trarre il corsivo e il neretto da stili che per loro natura non hanno tali varianti. Per ottenerlo si utilizzano tecniche di deformazione e di trascinamento. In generale sarebbe bene evitare di sfruttare tali possibilità, dal momento che se uno stile non dispone di una serie, significa che non è adatto per quella.

## 2.2 Tipometria

La tipometria è la misurazione degli elementi che riguardano la composizione e l'impaginazione. Le voci più importanti sono costituite dai corpi (l'altezza dei caratteri), dalla spaziatura, dall'interlinea, dalla giustezza e dalla giustificazione. In breve, il corpo è l'altezza del carattere, la spaziatura è la distanza tra una parola e l'altra in una riga, l'interlinea è lo spazio verticale aggiuntivo tra le righe, la giustezza è lo spazio orizzontale che le righe di testo hanno a disposizione, la giustificazione è il procedimento di regolazione della spaziatura e dell'interlinea in modo da ottenere un allineamento delle righe con i margini (sia in orizzontale che in verticale).

### 2.2.1 Corpo, dimensioni e scala

La dimensione del carattere si misura in senso verticale e si definisce ***corpo***. Per misurare il corpo e le altre dimensioni che riguardano i caratteri si possono utilizzare diverse unità di misura, ma quando si tratta di sistemi di composizione elettronica a mezzo di software, è molto probabile che si disponga solo del pica e del punto anglo-americano:

- 1 pica = 1/6 di pollice;
- 1 punto = 1/12 di pica = 1/72 di pollice.

Per comprendere cosa sia il corpo di un carattere è bene descrivere le varie componenti dell'altezza di questo. La figura 2.1 mostra schematicamente la parola «Agglomerato» suddivisa orizzontalmente secondo le componenti verticali della dimensione del carattere.

Figura 2.1. Le dimensioni del carattere.


Il carattere si appoggia su una linea che rappresenta la base della «parte mediana»; le lettere come la «l» si alzano occupando anche la «parte ascendente»; altre, come la «g», si allungano in basso a occupare la «parte discendente». Il corpo del carattere include anche uno spazio aggiuntivo: la «spalla». Si distingue una spalla superiore, che è uno spazio minimo sopra la parte ascendente, e la spalla inferiore, che si trova al di sotto della parte discendente (nella figura la spalla è molto grande, in proporzione, rispetto alla realtà).

La distanza tra la base di una riga (la base della parte mediana) e la base di quella successiva dovrebbe essere superiore o al minimo uguale alla grandezza del corpo. Quando questa distanza è superiore, lo spazio aggiuntivo è l'***interlinea***. Con i sistemi di composizione elettronica per mezzo di software, si misura generalmente lo spazio tra le basi delle righe ed è ammissibile anche l'utilizzo di distanze inferiori all'altezza del carattere, ottenendo in pratica una sovrapposizione della parte mediana inferiore di una riga con la parte mediana superiore di quella successiva.

La rappresentazione di un carattere con un corpo di una data dimensione dipende dalla disponibilità di questo. Con i sistemi tipografici tradizionali era necessario disporre di una serie di caratteri mobili differenti, distinti in base a una scala. Con i sistemi di composizione elettronica via software si possono trovare dei caratteri riproducibili in qualsiasi corpo, eventualmente generando dei file opportuni per la scala richiesta. Tuttavia, in presenza di dimensioni particolarmente piccole si rischia di perdere dei dettagli importanti dei segni che compongono lo stile utilizzato e,

di conseguenza, potrebbe essere preferibile l'utilizzo di una variante dello stile che sia più adatta alle dimensioni ridotte.

### 2.2.2 Giustezza, spaziatura e giustificazione orizzontale

La giustezza è lo spazio orizzontale a disposizione delle righe di testo; in altri termini, è la larghezza della colonna all'interno della quale si può distribuire il testo. La spaziatura è lo spazio tra la fine di una parola e l'inizio di quella successiva.

> Nei testi in italiano, la spaziatura è uniforme, senza eccezioni, a differenza della tradizione tipografica di altri paesi. Per esempio, la spaziatura dopo un punto fermo è esattamente uguale a quella di qualunque altra situazione. Quando si utilizza il sistema di composizione TeX per scrivere un testo in italiano, si dovrebbe inserire il comando '`\frenchspacing`' per evitare anomalie nella spaziatura.

Quando si vuole ottenere un allineamento del testo all'inizio e alla fine della giustezza, si parla di giustificazione (orizzontale). Per ottenerla, è necessario che la spaziatura sia adattata in modo da arrivare a questo risultato. La giustificazione orizzontale è solo una delle scelte stilistiche che il tipografo ha a disposizione: non si tratta di una convenzione obbligatoria.

### 2.2.3 Giustificazione verticale

Come nel caso della giustificazione orizzontale, ci può essere la necessità o l'opportunità di adattare l'interlinea in modo da riempire completamente le pagine. Ciò si ottiene attraverso la giustificazione verticale.

## 2.3 Il carattere nel software di composizione

Utilizzando i programmi di composizione tipografica si è costretti generalmente a fare i conti con la terminologia dei paesi di lingua inglese e con altri problemi legati alla rappresentazione simbolica dei segni all'interno del software. La tradizione tipografica di questi ha generato dei termini che non sono perfettamente traducibili con concetti della tradizione italiana, per cui si utilizzano alcuni termini di origine anglofona, eventualmente tradotti in modo letterale.

### 2.3.1 Terminologia

In inglese si utilizza normalmente il termine *font* per fare riferimento al carattere tipografico. Spesso non si traduce questo termine in qualcosa che riguardi la tradizione tipografica italiana, mantenendo piuttosto il termine inglese invariato; tuttavia, alle volte viene utilizzata la forma: ***fonte***.

> Se il contesto non richiede un'aderenza perfetta con il termine originale inglese, si possono usare forme espressive più comprensibili, come «carattere», «tipo di carattere», «carattere tipografico» o «carattere da stampa».

### 2.3.2 Caratteristiche di una carattere tipografico elettronico

Il carattere tipografico usato nel software applicativo di composizione, ha una serie di caratteristiche, alcune delle quali sono fondamentali.

- ***foundry*, *fonderia***

  La fonderia è il produttore del carattere tipografico, cioè chi ha creato la tipizzazione, pur senza esserne il disegnatore. Per fare un esempio comune, Adobe è la fonderia dello stile Times New Roman.

- ***family*, *famiglia***

  La famiglia del carattere, inteso come traduzione del termine *font family*, corrisponde simultaneamente alla specie e allo stile del carattere. In altri termini, rappresenta sia la specie alfabetica che lo stile. Per fare un esempio, la famiglia Times New Roman è un carattere di specie latina e di stile Times.

- All'interno di una famiglia si distinguono normalmente le serie riferite alla forma: spessore (*weight*), inclinazione (*slant*) e larghezza (*set*, o *width*).

- ***codifica***

  La codifica rappresenta l'elemento nuovo più importante nelle caratteristiche di un carattere tipografico per l'elaborazione via software. Il problema viene descritto nella prossima sezione.

### 2.3.3 Codifica

L'utilizzo dei caratteri con i sistemi di composizione basati sul software richiede un abbinamento tra segni e simboli binari. Questo abbinamento è definito dalla codifica. Il problema si può intendere meglio se si pensa a un programma a composizione differita.

In questi casi si parte da un file sorgente, scritto probabilmente secondo la codifica ISO 8859-1, con il quale il programma deve comporre il risultato, utilizzando i caratteri a disposizione.

Il carattere tipografico utilizzato dal programma di composizione è contenuto normalmente all'interno di file, da cui questo programma estrae le informazioni necessarie attraverso un riferimento dato da un codice numerico. In condizioni normali, il programma di composizione fa riferimento al simbolo binario utilizzato nel sorgente per ottenere il segno corrispondente all'interno del carattere tipografico utilizzato (eventualmente attraverso una qualche traslazione). In pratica, alla lettera «A» nel sorgente dovrebbe corrispondere la lettera «A» del carattere tipografico che si sta utilizzando, ma se il carattere tipografico è organizzata in modo differente, si potrebbe ottenere qualcosa di diverso. Questo problema si avverte di solito quando si utilizza una famiglia di caratteri che fa riferimento a una specie simbolica, o comunque a un alfabeto che non ha alcuna corrispondenza con la codifica utilizzata nel sorgente. In questi casi, di solito, per rappresentare i segni si può fare uso di comandi speciali interpretati opportunamente dal programma di composizione.

Un programma di composizione potrebbe disporre di caratteri tipografici che hanno solo una corrispondenza parziale con la codifica utilizzata per scrivere il sorgente, per esempio, potrebbero mancare alcuni segni che vengono messi a disposizione attraverso altre specie.

## 2.4 Problemi legati ai caratteri

Nelle origini della tipografia, molti caratteri mobili rappresentavano l'unione di più lettere o altri segni in logotipo (cioè l'unione in un simbolo unico). L'unione di questi derivava da delle consuetudini stilistiche o dalla forma dei segni adiacenti che per qualche motivo potevano richiedere un avvicinamento o un adattamento.

Il ***legato*** (in inglese *ligature*) è l'unione di due o più segni per motivi storici o estetici; i più comuni sono le sequenze «fi», «fl» e «ffi», dove le lettere vengono avvicinate in modo particolare fino a unirsi o a inglobarsi. Alcune forme di legato si sono tradotte in segni indipendenti, come nel caso di «AE» che si è trasformato in «Æ», «sz» che nella lingua tedesca è ormai «ß», «et» (latino) che è divenuto «&», ovvero l'attuale e-commerciale, e anche «ad» (latino), che nella lingua inglese è diventato «@» (*at*).

L'avvicinamento delle lettere, era ed è motivato dalla forma di queste, per evitare il formarsi di vuoti visivi che potrebbero creare difficoltà alla lettura. I casi più comuni sono le sequenze «AV», «AT», «AY».

## 2.5 Riferimenti

- *Scienza, tecnologia e arte della stampa e della comunicazione*, Arti poligrafiche europee

  <*http://www.apenet.it/grafica/libri/Grafica/Grafica01/indice02.html*>

  <*http://www.apenet.it/grafica/libri/Grafica/Grafica02/indice02.html*>

  <*http://www.apenet.it/grafica/libri/Grafica/Grafica03/indice02.html*>

- *Scienza, tecnologia e arte della stampa e della comunicazione*: Giuseppe Pellitteri, Luigi Farinelli, *Grafismi*, Arti poligrafiche europee

  <*http://www.apenet.it/grafica/libri/Grafica/Grafica01/1123.html*>

# Stile letterario

Questo capitolo, vuole essere solo un riferimento essenziale alla definizione di uno stile letterario e il contenitore di una piccola raccolta di regole, che dovrebbero semplificare la vita di chi scrive documenti elettronici.

L'autore di questo documento non ha una competenza specifica su questo problema; tuttavia, è importante almeno affrontare l'argomento sottolineando alcuni concetti importanti.[1]

## 3.1 Uniformità

Il concetto di ***stile letterario*** potrebbe essere espresso semplicemente spiegando l'esigenza di realizzare un documento ***uniforme***: sia dal punto di vista visivo, sia dal punto di vista espressivo. Questo coinvolge quindi l'aspetto grammaticale (ortografia, sintassi, lessico, ecc.) e l'aspetto tipografico (impaginazione, tipi di carattere, dimensione, ecc.) o artistico.

L'esigenza di un'uniformità visiva deriva dal piacere e dal rilassamento che può dare al lettore un documento impaginato e strutturato in un modo ordinato e chiaro, per la facilità nella lettura che ne deriva. Nello stesso modo è importante l'uniformità grammaticale, cosa particolarmente delicata in una lingua come quella italiana in cui sono consentite molte variazioni, data la varietà linguistico-culturale delle varie regioni.

Il novello scrittore di documentazione tecnica, che scrive e impagina senza l'aiuto di un editore, tende a comprendere l'esigenza di uno stile tipografico, dimenticando che esiste anche uno stile espressivo-grammaticale.

Il problema dell'uniformità stilistica si accentua quando si deve collaborare alla realizzazione di un progetto letterario. L'uniformità non è più solo un fatto di coerenza personale, ma di coerenza complessiva di tutto il gruppo.

La coordinazione dei vari collaboratori è un problema delicato e diviene essenziale la stesura di uno standard letterario complessivo. Alle volte questo ferisce la sensibilità di alcuni collaboratori e genera discussioni senza fine e senza soluzione.[2]

## 3.2 Regole di composizione del testo

Il modo migliore per definire uno stile grammaticale è lo studio su un testo di grammatica. Qui si vogliono solo raccogliere alcuni punti essenziali che non possono essere ignorati. In effetti, il tipico autore di testi a carattere tecnico, specialmente quando non si tratta di un'attività professionale remunerata, ha un'ottima conoscenza dell'argomento trattato e una pessima padronanza della lingua.

### 3.2.1 Punteggiatura e spaziatura

La punteggiatura si compone di quei simboli che consentono di separare le parole e di delimitare le frasi.

[1] Come sempre, tutte le segnalazioni di errore sull'ortografia, la sintassi e il contenuto di questo documento, sono gradite. :-)

[2] Il vero artista è colui che crea qualcosa di nuovo e non accetta di sottostare alle regole generali. È evidente quindi che questa persona non potrà lavorare in un gruppo perché non si sottometterà mai alle regole poste dagli altri o dalla consuetudine.

- Ogni parola è separata da un solo spazio.

  Tipograficamente, lo spazio è una separazione di ampiezza non definita, spesso ampliato o compresso, per ottenere un allineamento del testo sia a sinistra che a destra. Un autore non deve pensare a queste cose quando scrive la propria opera; si deve limitare a spaziare le parole con un solo carattere spazio.[3]

  La dattilografia insegnava a ottenere testi allineati a sinistra e a destra con l'inserzione opportuna di spazi aggiuntivi, vicino alle parole composte da poche lettere (congiunzioni, articoli, ecc.). Questo tipo di tecnica è ormai da abbandonare, lasciando semmai che siano i programmi di composizione a prendersi cura di questi problemi, anche quando il risultato finale deve essere un file di testo puro e semplice.

  I programmi di composizione più evoluti facilitano il compito dello scrittore eliminando gli spazi superflui, per cui con questi non c'è l'esigenza di porre attenzione alla dimensione delle spaziature.[4]

- I simboli di punteggiatura normale sono attaccati alla parola che precede e separati con uno spazio dalla parola che segue.

  Si tratta di: punto, virgola, due punti, punto e virgola, punto interrogativo e punto esclamativo.

  Alle volte, l'autore di documenti tecnici di informatica si lascia confondere dall'uso che si fa di tali simboli in un particolare linguaggio di programmazione o in altri ambiti analoghi. È chiaro, per esempio, che se si deve indicare un'estensione di un file, come «.sgml», non si può rispettare tale regola, ma il punto che precede quell'estensione non rappresenta un simbolo di punteggiatura del testo.

- Le parentesi sono attaccate al testo che racchiudono e, rispetto alla punteggiatura esterna, si comportano come un'unica parola.

  La parentesi di apertura è separata con uno spazio dalla parola che precede, mentre quella di chiusura è separata con uno spazio dalla parola che segue. I simboli di punteggiatura normale che dovessero seguire una parentesi chiusa vanno attaccati a questa ultima.

  Nella lingua italiana non è consentito racchiudere all'interno di parentesi un periodo terminante con il punto fermo. Questa modalità è tipica della lingua inglese e i traduttori devono tenerne conto, al limite togliendo le parentesi nella frase tradotta.

- Il testo riportato tra virgolette si comporta come quello racchiuso tra parentesi.

  La lingua italiana prevede l'uso di virgolette uncinate (in basso), virgolette elevate doppie e singole. Secondo la grammatica, le virgolette uncinate, o virgolette basse, sono da preferire. Tuttavia, dal momento che le virgolette elevate possono essere ottenute anche utilizzando soltanto il codice ASCII tradizionale a 7 bit, molti autori preferiscono accontentarsi e utilizzare solo quelle elevate.[5]

- Il trattino di unione è corto e unito alle parole da collegare.

[3] Secondo una regola della tipografia del passato, ormai condannata generalmente, era necessario aumentare lo spazio che divide la fine di un periodo dall'inizio del successivo. Per qualche ragione si trovano ancora documenti in lingua inglese che seguono questa regola, anche quando si tratta di file di testo.

[4] Purtroppo LaTeX segue la vecchia regola dell'allungamento dello spazio dopo il punto fermo che chiude il periodo, con l'aggravante che per riuscire a determinarlo può fare solo delle supposizioni, che a volte sono errate. Per fare in modo che LaTeX eviti di applicare questa regola errata, si può utilizzare il comando '`\frenchspacing`' nel preambolo del documento.

[5] Quando il sistema di composizione si basa su TeX e si usano virgolette elevate, le virgolette doppie si ottengono preferibilmente attraverso una coppia di apici singoli aperti (‘‘‘’) e una coppia di apici singoli chiusi (‘’’’). In altri casi, soprattutto quando si tratta di file di testo puri e semplici, gli apici doppi si indicano con le virgolette normali (‘"…"’).

Si usa per unire insieme due parole in modo da formare una parola composta. I programmi di composizione tendono a considerare un trattino singolo come un trattino corto, proprio per questo scopo.

- La lineetta, o trattino lungo, serve per introdurre un discorso diretto, oppure un inciso.

  Il trattino utilizzato per delimitare un discorso diretto, viene usato normalmente solo in apertura. Può apparire anche un trattino in chiusura quando al discorso diretto segue un commento. Se il trattino si usa per delimitare un inciso, si usa per aprirlo e solitamente anche per chiuderlo, come se si trattasse di parentesi.

  Generalmente, il trattino lungo è preceduto e seguito da uno spazio; davanti al trattino di chiusura vanno collocati il punto interrogativo, il punto esclamativo e i puntini, mentre per gli altri simboli di punteggiatura non esiste una convenzione precisa.[6]

### 3.2.2 Utilizzo dei simboli di interpunzione

L'uso della punteggiatura nella lingua italiana è definito da regole molto vaghe che si prestano a facili eccezioni di ogni tipo. Qui si elencano solo alcuni concetti fondamentali.

- ,

  La virgola è un segno di interpunzione che collega due segmenti di testo separati da una pausa debole.

- ;

  Il punto e virgola è un segno di interpunzione che si colloca a metà strada tra la virgola e il punto. Non segna la chiusura di un periodo.

- :

  I due punti sono un simbolo di interpunzione ***esplicativo***. Collegano due segmenti di testo separati dal punto di vista sintattico, in cui la seconda parte, quella che segue il simbolo, elenca, chiarisce o dimostra il concetto espresso nella prima parte.

- .

  Il punto fermo è un segno di interpunzione che collega due segmenti di testo separati da un pausa forte. Generalmente segna la conclusione di un periodo. La parola successiva al punto ha l'iniziale maiuscola.

- !

  Il punto esclamativo indica generalmente la conclusione di un'esclamazione affermativa. Generalmente, quando conclude un periodo, il testo che segue ha l'iniziale maiuscola.

- ?

  Il punto di domanda indica un tono interrogativo alla fine di una frase. Generalmente, quando conclude un periodo, il testo che segue ha l'iniziale maiuscola.

- ...

  I punti di sospensione sono in numero fisso di tre e indicano che il discorso non viene portato a conclusione. Generalmente, sono uniti alla parola o al segno di interpunzione che

[6] TeX permette l'uso di tre trattini di lunghezza differente: il trattino corto che si ottiene con un trattino singolo, il trattino medio che si ottiene con due trattini in sequenza e il trattino lungo che si ottiene con tre. Nella lingua italiana vanno usati solo i primi due, dove il trattino medio di TeX corrisponde al trattino lungo della grammatica italiana.

li precede, oppure distanziati, a seconda che siano solo una sospensione oppure indichino l'omissione di un nome o di un'altra parola.

Se si trovano alla fine di un periodo, dove andrebbe collocato un punto, questo non viene aggiunto e la frase successiva inizia con la maiuscola. Nello stesso modo, se si trovano alla fine di un'abbreviazione che termina con un punto, questo punto viene assorbito.

- ecc.

  Il punto di abbreviazione, quando si trova alla fine di un periodo, conclude da solo anche il periodo stesso, ed è seguito da iniziale maiuscola.

- ( )

  Le parentesi, generalmente tonde, servono per delimitare un inciso, come un commento, una nota dello scrivente, un chiarimento, ecc. Generalmente, i commenti del redattore o del traduttore sono terminati, entro l'ambito delle parentesi, con le sigle NdR (nota del redattore) e NdT (nota del traduttore).

### 3.2.3 Accenti e troncamenti

Nella lingua italiana scritta, l'uso degli accenti è un fatto puramente convenzionale. Ciò significa che l'accento non indica necessariamente il suono che ha effettivamente la lettera accentata, ma solo la sua rappresentazione consueta (più avanti, nella sezione 3.2.4 è riportato il testo originale della norma UNI 6015 sul «segnaccento obbligatorio»).[7]

- Nella lingua scritta è prevista (ed è obbligatoria) solo l'accentazione delle vocali finali delle parole nelle quali il tono della voce si rafforza sull'ultima sillaba (accento grafico).

  È possibile l'uso dell'accento per le vocali interne quando ciò serva per togliere ambiguità tra termini omografi (scritti nello stesso modo) che abbiano significati differenti. Generalmente, questa ambiguità è risolta dal contesto e raramente si incontra la necessità di utilizzare accenti interni.

- Si utilizza comunemente solo l'accento grave (**àèìòù**), con l'eccezione della vocale «e» che può avere l'accento acuto (**é**).

- Vogliono l'accento acuto le parole terminanti in **ché** (perché, poiché, ecc.), oltre a **né** (congiunzione) e **sé** (pronome tonico). In particolare, **sé** viene scritto generalmente senza accento quando è seguito da **stesso**, anche se la grammatica non lo richiede.

- Vogliono l'accento alcuni monosillabi contenenti due vocali: **ciò**, **già**, **giù**, **più** e **può**.

- Vogliono l'accento i monosillabi che senza potrebbero avere un significato differente. La tabella 3.1 mostra l'elenco dei monosillabi accentati più importanti.

- Non vogliono l'accento alcuni monosillabi tra cui: **qui**, **qua**, **sto** e **sta**.

- Solo alcune parole tronche richiedono la segnalazione di tale troncamento con l'apostrofo finale. In particolare: **po'** (poco), **mo'** (modo), **ca'** (casa) e alcuni imperativi.

- L'accento circonflesso (^) non si usa più. Serviva per i nomi terminanti in **-io** che al plurale terminerebbero in **-ii** (per esempio: armadio, armadii). Attualmente, si tende a usare questi plurali con una sola **-i** finale, a parte i casi in cui ciò genera ambiguità (**assassino**, **assasini**; **assassinio**, **assassinii**).

[7] Nell'ambito della documentazione tecnica, sarebbe consigliabile di evitare l'uso di accentazioni non comuni, anche se queste potrebbero essere preferibili in contesti più raffinati.

Tabella 3.1. Elenco dei monosillabi accentati più importanti e dei loro equivalenti (omografi) non accentati.

| | | | |
|---|---|---|---|
| dà | indicativo di dare (dà valore) | da | preposizione (da voi) |
| è | verbo | e | congiunzione |
| là | avverbio (resta là) | la | articolo |
| lì | avverbio (vado lì) | li | pronome |
| né | congiunzione (né questo né quello) | ne | pronome (ne voglio ancora) |
| sé | pronome tonico (pieno di sé) | se | pronome atono o congiunzione |
| sì | avverbio (dice di sì) | si | pronome |

Alle volte, l'uso delle vocali accentate può creare problemi tecnici, dovuti alla loro mancanza nell'insieme di caratteri a disposizione. In Italia, come nei paesi dell'Europa centrale, si utilizza la codifica ISO 8859-1 (Latin 1) che contiene tutte le lettere accentate necessarie. Nelle circostanze in cui ciò non è attuabile (per esempio quando si dispone di un sistema configurato male, o la tastiera non dispone dei simboli necessari), occorre utilizzare delle tecniche di rappresentazione che dipendono dal programma utilizzato per la composizione.

SGML e XML, comprendendo in queste categorie anche HTML e XHTML, dispongono di una serie di entità standard, a cui corrispondono in particolare le macro elencate nella tabella 3.2.

Tabella 3.2. Vocali accentate attraverso l'uso di macro SGML e XML.

| Vocale accentata | Macro corrispondente |
|---|---|
| à, À | à, À |
| è, È | è, È |
| ì, Ì | ì, Ì |
| ò, Ò | ò, Ò |
| ù, Ù | ù, Ù |
| é, É | é, É |

TeX (e di conseguenza LaTeX) dispone di una serie di codici elencati nella tabella 3.3.

Tabella 3.3. Vocali accentate per TeX.

| Vocale accentata | Codice TeX corrispondente |
|---|---|
| à, À | \‘a, \‘A |
| è, È | \‘e, \‘E |
| ì, Ì | \‘{\i}, \‘I |
| ò, Ò | \‘o, \‘O |
| ù, Ù | \‘u, \‘U |
| é, É | \’e, \’E |

Lout dispone del comando ‘**@Char**’ per indicare simbolicamente i segni tipografici che per qualche ragione non possono essere scritti letteralmente attraverso la codifica a disposizione. La tabella 3.4 mostra i comandi necessari a ottenere le vocali accentate.

Tabella 3.4. Vocali accentate per Lout.

| Vocale accentata | Comando di Lout |
|---|---|
| à, À | @Char agrave, @Char Agrave |
| è, È | @Char egrave, @Char Egrave |
| ì, Ì | @Char igrave, @Char Igrave |
| ò, Ò | @Char ograve, @Char Ograve |
| ù, Ù | @Char ugrave, @Char Ugrave |
| é, É | @Char eacute, @Char Eacute |

Quando si scrive un file di testo puro e semplice, ma non è possibile utilizzare la codifica ISO 8859-1, si può aggiungere un apice opportuno subito dopo la vocale da accentare. Naturalmente questa tecnica può valere solo per la lingua italiana in cui gli accenti si pongono solo nelle vocali finali. Visivamente il risultato è molto simile a quello corretto.

Tabella 3.5. Trucco per rappresentare le vocali accentate quando non si può fare altrimenti.

| Vocale accentata | Vocale apostrofata corrispondente |
|---|---|
| à, À | a‘, A‘ |
| è, È | e‘, E‘ |
| ì, Ì | i‘, I‘ |
| ò, Ò | o‘, O‘ |
| ù, Ù | u‘, U‘ |
| é, É | e’, E’ |

### 3.2.4 Segnaccento obbligatorio (UNI 6015)

Quello che segue è la norma UNI 6015 sull’uso degli accenti. Il testo è stato ottenuto da *Scienza, tecnologia e arte della stampa e della comunicazione, Preparazione del manoscritto* <*http://www.apenet.it/grafica/libri/Grafica/Grafica01/1206.html*>.

Segnaccento obbligatorio nell’ortografia della lingua italiana (Uni 601567):

1. *Scopo*

La presente unificazione ha lo scopo di stabilire le regole ortografiche per il segnaccento nei testi stampati in lingua italiana, quando esso sia obbligatorio.

2. *Definizione*

2.1 Il segnaccento (o segno d’accento, o accento scritto) serve a indicare esplicitamente la vocale tonica, per esempio: *andrà*, *colpì*, *temé*, *virtù*.

2.2. Il segnaccento può essere grave (‘’’) o acuto (‘’’).

3. *Uso*

Il segnaccento è obbligatorio nei casi seguenti:

3.1. Su alcuni monosillabi, per distinguerli da altri monosillabi che si scrivono con le stesse lettere ma senza accento:

*ché* («poiché», congiunzione causale) per distinguerlo da *che* (congiunzione in ogni altro senso, o pronome);

*dà* (indicativo presente di dare) per distinguerlo da *da* (preposizione) e *da’* (imperativo di dare);

*dì* («giorno») per distinguerlo da *di* (preposizione) e *di'* (imperativo di dire);

*è* (verbo) per distinguerlo da *e* (congiunzione);

*là* (avverbio) per distinguerlo da *la* (articolo, pronome, nota musicale);

*lì* (avverbio) per distinguerlo da *li* (articolo, pronome);

*né* (congiunzione) per distinguerlo da *ne* (pronome, avverbio);

*sé* (pronome tonico) per distinguerlo da *se* (congiunzione, pronome atono);

*sì* («così», o affermazione) per distinguerlo da *si* (pronome, nota musicale);

*té* (pianta, bevanda) per distinguerlo da *te* (pronome).

3.2. Sui monosillabi: *chiù, ciò, diè, fé, già, giù, piè, più, può, scià*.

3.3. Su tutte le parole polisillabe su cui la posa della voce cade sulla vocale che è alla fine della parola, per esempio: *pietà*, *lunedì*, *farò*, *autogrù*.

4. *Forma*

4.1. Il segnaccento, nei casi in cui è obbligatorio, è sempre grave sulle vocali: *a*, *i*, *o*, *u*.

4.2. Sulla *e*, il segnaccento obbligatorio è grave se la vocale è aperta, è acuto se la vocale è chiusa:

- è sempre grave sulle parole seguenti:

*ahimè* e *ohimè*, *caffè*, *canapè*, *cioè*, *coccodè*, *diè* e *gilè*, *lacchè*, *piè*, *tè*; inoltre sulla maggior parte dei francesismi adattati, come *bebè*, *cabarè*, *purè*, ecc. e sulla maggior parte dei nomi propri, come *Giosuè*, *Mosè*, *Noè*, *Salomè*, *Tigrè;*

- è acuto sulle parole seguenti:

*ché* («poiché») e i composti di *che* (*affinché*, *macché*, *perché*, ecc.), *fé* e i composti *affé*, *autodafé*, i composti di *re* e di *tre* (*viceré*, *ventitré*), i passati remoti (*credé*, *temé*, ecc., escluso *diè*), le parole *mercé*, *né*, *scimpanzé*, *sé*, *testé*.

4.3. Anche per la *o* si possono distinguere i due timbri (aperto o chiuso) con i due accenti (grave ed acuto) ma solo in casi in cui l'accento è facoltativo, per esempio: *còlto* (participio passato di *cogliere*), e *cólto* («istruito»).

### 3.2.5 Uso della «d» eufonica

Le congiunzioni **e**, **o** e la preposizione **a**, consentono l'aggiunta di una **d** eufonica, per facilitarne la pronuncia quando la parola che segue inizia per vocale. Si tratta di una possibilità e non di una regola; di questa **d** si potrebbe benissimo fare a meno.

Ognuno tende a usare questa **d** eufonica in modo differente, a seconda della propria cadenza personale, che ne può richiedere o meno la presenza. Quando si scrive, bisognerebbe mantenere lo stesso stile, anche sotto questo aspetto, quindi ognuno deve stabilire e seguire un proprio modo.

Esiste tuttavia un suggerimento che punta all'uso moderato di queste **d** eufoniche: usare la **d** solo quando la vocale iniziale della parola successiva è la stessa; e non usarla nemmeno quando, pur essendoci la stessa vocale iniziale nella parola successiva, ci sia subito dopo una **d** che possa complicare la pronuncia.

### 3.2.6 Elisione davanti alla lettera «h»

In linea di massima, l'articolo che si mette davanti a un termine che inizia con la lettera **h**, è quello che si userebbe pronunciando quella parola come se iniziasse per vocale. Secondo questo principio, va usata l'elisione, così come si fa con i termini che iniziano per vocale, senza alcuna «h» anteriore. Per esempio: l'harem; l'hotel; l'host.

Tuttavia, quando si tratta di un termine che, proveniendo da un'altra lingua, non è ancora diventato di uso comune e nella lingua originale si pronuncia con la lettera «h» iniziale aspirata, si preferisce evitare l'elisione.

### 3.2.7 Uso delle maiuscole

L'iniziale maiuscola si utilizza all'inizio del periodo e per evidenziare i nomi propri. Nel dubbio è meglio evitare di utilizzare le maiuscole. La lingua italiana fa un uso diverso delle maiuscole rispetto ad altre lingue. Il novello scrittore di documenti tecnici tende a lasciarsi influenzare dall'uso che si fa delle maiuscole nella lingua inglese. Per questo è bene ribadire che in italiano l'uso di queste deve essere ridotto al minimo indispensabile.

### 3.2.8 Plurali

Ci sono alcuni aspetti del plurale nella lingua italiana che vale la pena di annotare. In particolare, nel caso di chi deve utilizzare anche termini stranieri, si pone il problema di decidere se questi siano invariabili o meno. A questo proposito, esistono due regolette semplici e pratiche:

- le parole terminanti per consonante sono invariate al plurale;
- i termini di provenienza straniera non ancora assimilati sono invariati al plurale.

In particolare, per quanto riguarda la seconda, la logica è che non si può applicare un plurale secondo le regole di una lingua straniera mentre si usa l'italiano. Inoltre, dato che nella maggior parte dei casi si tratta di termini inglesi, che nella loro lingua prenderebbero quasi sempre una terminazione in **-s** al plurale, diventerebbe anche difficile la loro pronuncia in italiano.

#### 3.2.8.1 Interfacce o interfaccie?

Esiste una regoletta che permette di stabilire facilmente come debba essere ottenuto il plurale delle parole che terminano in **-cia** e **-gia**: la **i** rimane se la **c** e la **g** sono precedute da vocale, oppure se la **i** viene pronunciata con accento, mentre viene eliminata se queste consonanti sono precedute da un'altra consonante.

Quindi si ha: **camicia**, **camicie** e **interfaccia**, **interfacce**; **ciliegia**, **ciliegie** e **spiaggia**, **spiagge**; **energia**, **energie**.

### 3.2.9 Elenchi

Gli elementi puntati, o numerati, possono essere composti da elementi brevi, oppure da interi periodi. Se tutti gli elementi sono brevi:

- l'elenco deve essere introdotto da una frase terminante con due punti;
- ogni elemento deve essere terminato con un punto e virgola, a eccezione dell'ultimo che termina normalmente con un punto.

La descrizione appena fatta mostra un esempio di elenco del genere. Se anche uno solo degli elementi è troppo lungo, è bene trasformare tutti gli elementi in periodi terminati da un punto. In tal caso, se l'elenco viene introdotto da una frase, anch'essa termina con un punto.

Ci possono essere situazioni in cui queste indicazioni non sono applicabili: come sempre è necessario affidarsi al buon senso.

### 3.2.10 Citazioni

Le citazioni, cioè le frasi o i brani riprodotti letteralmente da altri documenti, devono apparire distinte chiaramente dal testo normale. Si usano normalmente queste convenzioni:

- quando la citazione è incorporata nel testo viene delimitata attraverso le virgolette, oppure utilizzando il corsivo se la citazione è particolarmente breve;
- le citazioni incluse in un'altra citazione già virgolettata si evidenziano attraverso l'uso di un altro tipo di virgolette, cominciando da quelle uncinate («»), utilizzando poi quelle elevate doppie (‘‘’’) e terminando con quelle singole (‘’);
- quando la citazione è molto lunga e occupa diversi capoversi, conviene utilizzare un corpo minore o un altro espediente tipografico per distinguerla dal testo normale, come con l'uso di rientri differenti;
- quando la citazione è lunga e non si vogliono utilizzare altri espedienti per evidenziarla, si utilizzano le virgolette, ripetendo quelle di apertura all'inizio di ogni capoverso;
- all'interno delle citazioni possono apparire dei commenti o chiarimenti inseriti da chi scrive, delimitandoli attraverso l'uso di parentesi quadre;
- all'interno delle citazioni vanno indicate le omissioni, che possono essere segnalate attraverso l'uso dei puntini di sospensione racchiusi tra parentesi quadre (come per i commenti);
- quando si fanno delle omissioni nella citazione all'inizio o alla fine del brano, è preferibile l'uso dei puntini di sospensione senza che questi siano racchiusi tra parentesi quadre; all'inizio i puntini di sospensione sono staccati dalla prima parola, mentre alla fine sono attaccati all'ultima.

## 3.3 Traduzioni e termini stranieri

Le traduzioni rappresentano un problema in più, dal punto di vista dell'uniformità stilistica espressiva, soprattutto perché sono frequentemente il risultato di un lavoro di gruppo. Il problema più grave è rappresentato dalla traduzione o dall'acquisizione di quei termini che non fanno parte del linguaggio comune.

- Una traduzione non può essere letterale, perché lingue diverse hanno strutture differenti e il significato che si attribuisce alle parole dipende dal contesto. Quello che conta, quindi, è che il significato sia mantenuto.
- Quando si tratta di termini tecnici di origine straniera, la loro traduzione può essere inopportuna, soprattutto quando chi deve esprimersi con quei concetti utilizza già abitualmente il termine in questione, nella forma originale, senza tradurlo.

  In pratica, è importante che gli utenti esperti possano trovare familiare la traduzione di un documento tecnico rivolto a loro.
- Una traduzione utilizzata largamente sul campo deve essere privilegiata al momento della scelta. È importante evitare che gli utenti esperti possano essere confusi da una traduzione. In pratica: gli utenti esperti dovrebbero trovare familiari le traduzioni scelte.
- Quando un termine straniero ha un significato più specifico della sua traduzione letterale, allora non conviene tradurlo.

  L'esempio più importante che deriva da questa affermazione è il termine **file**, che in italiano identifica precisamente il concetto di ***archivio elettronico generico***.

L'attività di traduzione è tanto più delicata se si considerano i vincoli posti dalle convenzioni internazionali che regolano l'editoria. In breve, la traduzione deve essere autorizzata dall'autore originale, verso il quale ci si assume la responsabilità del buon esito di questa operazione.

Per questo, la traduzione non può alterare il contenuto espresso dall'autore originale e nemmeno chiarirlo. Nello stesso modo, una traduzione deve sempre essere accompagnata dall'indicazione dei nomi dei traduttori che l'hanno realizzata.

### 3.3.1 Acquisizione di termini inglesi

Quando si decide di lasciare inalterato il termine straniero nel testo italiano, si pone il problema di stabilire il modo con cui questo possa convivere con il resto del testo. L'unica regola sicura è la verifica dell'uso generale, attraverso la discussione nelle liste specializzate. Tuttavia si possono definire alcune regole di massima, per dare l'idea del problema.

È importante osservare che nell'ambito delle traduzioni di documenti tecnici, nella stragrande maggioranza dei casi, si ha a che fare con l'inglese. Infatti, l'acquisizione di un termine straniero tende a seguire logiche differenti a seconda della lingua di origine. Per comprenderlo basta pensare con quanta facilità si potrebbe acquisire un termine francese, come «console», rispetto a un termine inglese.

- La prima cosa da fare di fronte a un termine da non tradurre è di verificare in un vocabolario di lingua italiana; se c'è, il problema è risolto. Questo potrebbe sembrare un consiglio banale; ma attualmente appaiono già parole come «input» e «output» che non sono poi di uso così generalizzato.

- Un termine inglese può assumere il genere che avrebbe se tradotto in italiano, oppure quello che suona meglio dandogli un significato italiano. In caso di dubbio è importante controllare l'uso comune (se esiste).

- I termini inglesi non tradotti sono invariabili al plurale, cioè quando sono inseriti in testi in italiano vanno scritti sempre al singolare, senza aggiungere la lettera «s» finale, anche se ci si riferisce a una quantità maggiore di uno.

  A titolo di esempio si pensi al termine «mouse» che al plurale inglese diventa «mice». Chi usa questo termine, probabilmente è costretto a farlo, dato che l'italiano offre poche alternative; forse si potrebbe indicare come «dispositivo di puntamento», ma questa definizione è troppo generica e probabilmente non verrebbe compresa. Pertanto, chi usa questi termini non può essere anche costretto a conoscere perfettamente l'inglese e il modo corretto di usare i plurali in quella lingua.

  In altri termini, la lingua italiana non può incorporare le regole di un'altra lingua.

Quando il termine che non si traduce non è di uso comune nell'ambiente a cui si rivolge il documento, dovrebbe essere evidenziato in corsivo tutte le volte che viene utilizzato. Per chiarire meglio il concetto, un termine tecnico può essere o meno di uso comune per il pubblico di lettori a cui si rivolge: se si tratta di un termine considerato normale per quell'ambiente, non è il caso di usare alcuna evidenziazione.

### 3.3.2 Stesura di un glossario

Quando si traduce un documento è importante la preparazione di un glossario, inteso come una raccolta di traduzioni standard che permettono di mantenere uniformità nel documento tradotto. Questo diventa tanto più importante quando si lavora in gruppo, o si partecipa alla traduzione di un gruppo di opere che fanno parte di uno stesso ambito tecnico.

Un glossario del genere non può essere un documento statico, in quanto si ha la necessità di aggiornare continuamente il suo contenuto; se non altro per estenderlo.

Nell'ambito della documentazione GNU, ci si può iscrivere alla lista `it@li.org` per chiedere informazioni sul lavoro già svolto e per discutere termini non ancora definiti dal glossario in corso di realizzazione. Per iscriversi basta inviare un messaggio a `majordomo@li.org` contenente nel corpo (e non nell'oggetto) il testo seguente:

'**`subscribe it`**'

L'invio di messaggi al gruppo di discussione va indirizzato poi a `it@li.org`.

Eventualmente si può scaricare il glossario attuale da <*ftp://ftp.linux.it/pub/People/md/glossario.tgz*>, tenendo presente che il moderatore della lista desidera che non sia distribuito ulteriormente, in modo da evitare che si diffondano versioni obsolete.

Come ultima nota è opportuno chiarire che un glossario per la traduzione può essere solo uno strumento, per l'utilizzo da parte di persone in grado di capire il contesto in cui i termini sono usati e di stabilire se le voci corrispondenti del glossario sono applicabili alle situazioni particolari.

### 3.3.3 Opere originali

Anche l'autore di un'opera originale di carattere tecnico, si imbatte in problemi simili a quelli dei traduttori. Infatti, quando l'acquisizione di un termine tecnico straniero riguarda solo l'ambito specializzato per il quale si scrive, si può dubitare del modo giusto di utilizzarlo.

Per questo, anche gli autori di opere originali possono avere la necessità di preparare un glossario e di discutere le espressioni migliori per un concetto determinato.

## 3.4 Strafalcioni comuni

L'influenza della lingua inglese porta a deformazioni sempre più frequenti nella lingua italiana. Queste annotazioni vogliono essere di aiuto a chi scrive in italiano sotto l'influenza della prosa inglese, sia perché sta traducendo, sia perché è abituato a leggere solo documentazione tecnica scritta in inglese. Il problema più evidente, ma più facile da affrontare, è quello dei «falsi amici»: quei termini che, pur assomigliandosi (e pur avendo, spesso, la stessa etimologia), hanno significati diversi nelle due lingue. Gli esempi più celebri sono «factory» che diventa erroneamente «fattoria» e «cold» che si trasforma in «caldo».

Il problema meno evidente e per questo più insidioso è dato dalle altre differenze fra le due lingue: la punteggiatura, l'uso delle maiuscole e la struttura delle frasi. Trascurando queste particolarità si rischia di ottenere un testo che è formalmente in italiano, ma che non «suona» come tale.

Per completare il quadro, viene mostrato qualche esempio comune per chiarire questi concetti, ma è bene ricordare che le possibilità sono infinite e che l'unico modo per scrivere in buon italiano è leggere tanto buon italiano (così come avviene per qualsiasi linguaggio di programmazione).

### 3.4.1 Falsi amici

I «falsi amici» sono quei termini inglesi che sembrano avere una traduzione ovvia in italiano, che però non è corretta. Lo specchietto che si vede nella tabella 3.6 mostra la traduzione corretta di alcuni termini, frequenti nei testi informatici, lasciando intuire l'errore comune che si fa al riguardo.

Tabella 3.6 Traduzioni corrette dei «falsi amici».

| | |
|---|---|
| consistent | coerente |
| exhaustive | esauriente |
| line | riga (quasi sempre) |
| re... (recursive) | ri... (ricorsivo) |
| set | insieme («set» è tennistico) |
| to set | impostare («settare» è di pessimo gusto) |
| subject | oggetto (di una lettera o di un messaggio) |
| to process | elaborare |
| to assume | supporre |
| proper (agg.) | giusto, corretto |
| proper (avv.) | vero e proprio |
| to support | si usi, per quanto possibile, una perifrasi |
| to return something | restituire qualcosa («ritornare» è intransitivo) |

### 3.4.2 Ortografia e sintassi

Quello che segue è un elenco di annotazioni riguardo all'uso dell'ortografia e della sintassi.

- La «e» o la «o» che introduce l'ultimo termine di un elenco non va preceduta da virgola. In inglese americano la norma è di usare la virgola (ma gli inglesi non la usano); a volte in italiano la virgola è ammissibile, ma si tratta di eccezioni.
- Se le frasi sono negative, allora devono essere separate con «né». Per esempio:

```
File che hanno questo bit settato non possono essere cancellati con DEL
o modificati.
```

  va sostituito con:

```
I file che hanno questo bit impostato non possono essere cancellati con
DEL né modificati.
```

- I periodi italiani sono più complessi di quelli inglesi, a parità di registro. Come buona regola, metà dei punti fermi vanno sostituiti con congiunzioni, subordinate, due punti o punti e virgola. L'esempio seguente di traduzione viene da *hostname(1)*.

```
-F, --file filename
        Read the host name from the specified file.  Comments
        (lines starting with a '#') are ignored.
```

```
-F, --file nomefile
        Legge il nome dell'host dal file specificato, ignorando
        i commenti (righe che iniziano con '#').
```

- L'uso del futuro in inglese è diverso da quello dell'italiano. L'esempio proviene da *mpage(1)*.

```
-O      Print 2 normal pages per sheet. But, this option
        will print every first and forth page of every set
        of four pages.  This option will ignore the -a and
        -l options.
```

```
-O      Stampa due pagine normali per foglio: questa opzione,
        però, stampa la prima e la quarta pagina per ogni dato
        insieme di quattro pagine. Questa opzione ignora le
        opzioni -a e -l.
```

- I nomi dei mesi sono minuscoli.
- I numeri (interi) che esprimono una quantità piccola vanno scritti preferibilmente per esteso.
- In italiano si usa, di solito, la sequenza nome+aggettivo; il contrario, aggettivo+nome, per quanto accettabile, ha spesso un significato diverso. Per esempio, si osservi la differenza tra «pover'uomo» e «uomo povero».
- Bisogna sempre concordare il genere grammaticale: «la directory padre» non ha senso.
- Spesso chi scrive in inglese usa contorsioni grammaticali assurde per evitare di denotare il genere della terza persona singolare; in particolare, si può trovare «they» o «their» usati al singolare: ovviamente in italiano ciò non va fatto. L'esempio proviene da *finger(1)*:

```
Mail status is shown as ‘‘No Mail.’’ if there is no mail at all, ‘‘Mail
last read DDD MMM ## HH:MM YYYY (TZ)’’ if the person has looked at their
mailbox since new mail arriving, or ‘‘New mail received ...’’, ‘‘
Unread since ...’’ if they have new mail.
```

## 3.5 Unità di misura

Nella documentazione a carattere scientifico diventa fondamentale la coerenza e la precisione nel modo in cui si indicano le grandezze e le unità di misura, oltre che la scelta di queste. In generale, ogni ambiente tecnico particolare tende a utilizzare le proprie grandezze e le proprie unità di misura, tralasciando gli sforzi di standardizzazione internazionale, contribuendo così a complicare inutilmente il proprio settore.

Purtroppo, l'ambito informatico costituisce l'esempio più problematico sotto questo aspetto, dal momento che l'esigenza di mantenere una compatibilità con il sistema binario ha attribuito a delle denominazioni ben precise del sistema decimale un significato differente rispetto a quello comune a tutti gli altri ambiti scientifici.

Lo standard internazionale sulle unità di misura è costituito dal SI, ovvero *Le Système international d'unités*, in italiano *Sistema internazionale di unità*. Il punto di riferimento per questo lavoro di armonizzazione è il BIPM (*Bureau international des poids et mesures*), con sede in Francia (<*http://www.bipm.fr/*>).

### 3.5.1 Come si scrive una grandezza

Per esprimere una quantità riferita a una grandezza in modo grafico, occorre disporre del ***simbolo*** (la sigla) che ne esprime l'unità di misura o un multiplo opportuno di tale unità, al quale si fa precedere il numero, in cifre, di tale quantità:

***n simbolo***

È importante che tra il numero e la sigla ci sia uno spazio, che non deve poter essere interrotto in fase di impaginazione del testo. Per esempio: si può scrivere 5 kg, ma non 5kg.

### 3.5.2 Nomi e simboli

È bene chiarire il significato di alcuni termini che riguardano la misurazione di qualcosa:

| | |
|---|---|
| grandezza | ciò che viene misurato, come la lunghezza, la massa, il tempo; |
| unità di misura | il nome attribuito a ciò che si usa per misurare, come il metro, il kilogrammo, il secondo; |
| simbolo | il simbolo che rappresenta l'unità di misura in modo standard, come «m», «kg», «s». |

Secondo il SI, il kilogrammo è l'unità di misura della massa, tenendo conto che i prefissi si utilizzano facendo riferimento al grammo. Si osservi inoltre che non si parla di «peso», perché questo termine è riferito piuttosto alla forza applicata a un oggetto.

I nomi delle unità di misura si esprimono generalmente senza iniziale maiuscola, mentre i simboli usati per rappresentarle simbolicamente vanno espressi esattamente come stabilito dagli standard, per quanto riguarda l'uso delle lettere maiuscole o minuscole.

Tabella 3.8. Esempi di grandezze e unità di misura.

| Grandezza | Unità di misura | Simbolo |
|---|---|---|
| lunghezza | metro | m |
| massa | kilogrammo | kg |
| tempo | secondo | s |
| corrente elettrica | ampere | A |

### 3.5.3 Prefissi moltiplicatori

Oltre alla definizione dei simboli che esprimono le unità di misura, si aggiungono dei simboli che rappresentano un multiplo ben preciso di tali unità. Tali simboli di moltiplicazione si pongono davanti al simbolo di unità a cui si riferiscono; per esempio, il simbolo «km» rappresenta mille unità «m», ovvero mille volte il metro.

I simboli che rappresentano tali moltiplicatori hanno anche un nome che normalmente si esprime senza iniziale maiuscola, indipendentemente dalla forma, maiuscola o minuscola, che ha il simbolo stesso.

I moltiplicatori riferiti alle unità di misura hanno un significato e un valore ben preciso. È un errore l'uso dei termini «kilo», «mega», «giga» e «tera», per rappresentare moltiplicatori pari a $2^{10}$, $2^{20}$, $2^{30}$ e $2^{40}$, come si fa abitualmente per misurare grandezze riferite a bit o a byte.

Tabella 3.9. Prefissi del *Sistema internazionale di unità* (SI).

| Nome | Simbolo | Valore | Note |
|---|---|---|---|
| yotta | Y | $10^{24}$ | |
| zetta | Z | $10^{21}$ | |
| exa | E | $10^{18}$ | |
| peta | P | $10^{15}$ | |
| tera | T | $10^{12}$ | |
| giga | G | $10^{9}$ | |
| mega | M | $10^{6}$ | |
| kilo | k | $10^{3}$ | Lettera «k» minuscola. |
| hecto, etto | h | $10^{2}$ | |
| deca | da | 10 | |
| | | 1 | Nessun moltiplicatore. |
| deci | d | $10^{-1}$ | |
| centi | c | $10^{-2}$ | |
| milli | m | $10^{-3}$ | |
| micro | $\mu$ | $10^{-6}$ | |
| nano | n | $10^{-9}$ | |
| pico | p | $10^{-12}$ | |
| femto | f | $10^{-15}$ | |
| atto | a | $10^{-18}$ | |
| zepto | z | $10^{-21}$ | |
| yocto | y | $10^{-24}$ | |

### 3.5.4 Prefissi per multipli binari

Lo standard IEC 60027-2 introduce un gruppo nuovo di prefissi da utilizzare in alternativa a quelli del SI, per risolvere il problema dell'ambiguità causata dall'uso improprio dei prefissi del SI in ambito informatico. A questo proposito, una discussione particolareggiata su questo argomento si può trovare nel documento *Standardized Units for Use in Information Technology*, di Markus Kuhn, <*http://www.cl.cam.ac.uk/~mgk25/information-units.txt*>.

Tabella 3.10. Prefissi IEC 60027-2.

| Origine | Nome | Simbolo | Valore | Note |
|---|---|---|---|---|
| kilobinary | kibi | Ki | $2^{10}$ | Si usa la «K» maiuscola. |
| megabinary | mebi | Mi | $2^{20}$ | |
| gigabinary | gibi | Gi | $2^{30}$ | |
| terabinary | tebi | Ti | $2^{40}$ | |
| petabinary | pebi | Pi | $2^{50}$ | |
| exabinary | exbi | Ei | $2^{60}$ | |
| zettabinary | zebi | Zi | $2^{70}$ | |
| yottabinary | yobi | Yi | $2^{80}$ | |

La tabella 3.10 riporta l'elenco di questi prefissi speciali.

## 3.6 Rappresentazione di valori

La rappresentazione di valori numerici tende a seguire forme differenti a seconda del contesto e delle convenzioni nazionali. Nella *Guide for the Use of the International Systems of Units (SI)*, pubblicato dal NIST (*National institute of standards and technology*), si trovano alcuni criteri per risolvere il problema in modo non ambiguo, validi anche al di fuori della realtà inglese.

### 3.6.1 Valori percentuali

In generale, l'uso del simbolo '**%**' va inteso come una forma abbreviata per 0,01 e in questo modo va usato, senza eccedere. In particolare, il simbolo di percentuale va posto dopo un valore numerico, staccato da questo, ma non separabile in fase di composizione tipografica:

```
n %
```

Per esempio, si può scrivere '$x$ **= 0,025 = 2,5 %**', mentre non è corretta la forma '$x$ **= 0,025 = 2,5%**'.

### 3.6.2 Valori numerici

Nella lingua italiana, come in molte altre, si usa la virgola come segno di separazione tra la parte intera e quella decimale, mentre nei paesi di lingua inglese, si utilizza il punto. A parte il problema di scegliere il segno opportuno in base alle proprie convenzioni nazionali, si pone piuttosto la difficoltà nel rappresentare numeri composti da una grande quantità di cifre.

La *Guide for the Use of the International Systems of Units (SI)* indica un metodo molto semplice e non equivoco: si separano le cifre a gruppi di tre, usando semplicemente uno spazio, sia prima che dopo il marcatore decimale, come si vede in questi esempi:

```
123 456 789
  3 456 789,012 345 6
      6 789,012 3
```

Naturalmente, lo spazio in questione non può consentire l'interruzione della riga in fase di composizione.

È ammissibile anche un'eccezione in presenza di raggruppamenti di sole quattro cifre, prima o dopo il marcatore decimale. In quel caso si può evitare la separazione:

```
1234
  23,2345
```

Un altro problema è quello della rappresentazione di valori numerici espressi con una base maggiore di 10, per i quali si utilizzano le prime 10 cifre numeriche e per il resto si usano le lettere alfabetiche. Queste lettere andrebbero utilizzate coerentemente, possibilmente in forma maiuscola.

## 3.7 Stile tipografico

La definizione dello stile tipografico è un altro punto delicato nella definizione dello stile letterario generale. Di solito, la sua preparazione, è compito del tipografo o del coordinatore di un gruppo di autori o traduttori.

Il modo migliore per stabilire e utilizzare uno stile tipografico è quello di usare un sistema SGML, attraverso cui definire un DTD che non permetta alcun dubbio nella relazione che ci deve essere tra le varie componenti di un documento. In questo modo, gli autori hanno solo il compito di qualificare correttamente le varie componenti del testo, senza pensare al risultato finale, per modificare il quale si può semmai intervenire sul sistema di conversione successivo.

Le sezioni seguenti trattano dei problemi legati alla definizione di uno stile tipografico per la redazione di documenti tecnico-informatici. L'idea è presa dalla guida di stile del gruppo di documentazione di Linux: LDP (*Linux documentation project*), ma le indicazioni si basano sulle consuetudini tipografiche italiane.

### 3.7.1 Blocchi di testo in generale

Scrivendo documenti che riguardano l'uso dell'elaboratore, si incorre frequentemente nella necessità di scrivere nomi, o intere parti di testo, che devono essere trattati in modo letterale. Possono essere nomi di file e directory, comandi, porzioni del contenuto di file, listati di programmi, ecc. In questi casi è sconsigliabile l'uso di un tipo di carattere proporzionale, perché si rischierebbe di perdere delle informazioni importanti. Si pensi al trattino utilizzato nelle opzioni della maggior parte dei comandi Unix: utilizzando un carattere proporzionale, attraverso un sistema di composizione come LaTeX, si otterrebbe un trattino corto, mentre due trattini posti di seguito genererebbero un trattino normale; e ancora, da tre trattini si otterrebbe un trattino largo.

### 3.7.2 Nomi di file e directory

I nomi di file, di qualunque tipo, dovrebbero essere rappresentati attraverso un tipo di carattere a spaziatura fissa.

I nomi di questi tipi di entità sono sensibili alla differenza tra maiuscole e minuscole. Per questo vanno scritti sempre così come sono, anche quando si trovano all'inizio di un periodo, senza acquisire un'eventuale iniziale maiuscola.

I nomi di file eseguibili, in quanto tali, sono indicati preferibilmente senza il percorso necessario al loro avvio.

I nomi di programmi per i sistemi Dos dovrebbero essere indicati utilizzando lettere maiuscole, senza tralasciare l'estensione.

### 3.7.3 Schermate, listati e simili

Il testo ottenuto da listati di vario tipo, come i pezzi di un programma sorgente, il risultato dell'elaborazione di un comando, o il contenuto di una schermata, possono essere rappresentati convenientemente attraverso un ambiente di inclusione di testo letterale a spaziatura fissa.

Il problema sta nel fatto che l'ampiezza di tale testo non può superare i margini del corpo del documento, in base al tipo di impaginazione finale che si ritiene dover applicare. Infatti, tale testo non può essere continuato nella riga successiva perché ciò costituirebbe un'alterazione delle informazioni che si vogliono mostrare.

Generalmente, non è possibile superare un'ampiezza di 80 colonne, pari a quella di uno schermo a caratteri normale.

### 3.7.4 Variabili di ambiente

I nomi di variabili di ambiente dovrebbero essere rappresentati attraverso un tipo di carattere a spaziatura fissa.

I nomi di questi tipi di entità sono sensibili alla differenza tra maiuscole e minuscole. Per questo vanno scritti sempre così come sono, anche quando si trovano all'inizio o all'interno di un periodo.

A seconda del tipo di documentazione, potrebbe essere stata definita la convenzione per cui questi nomi debbano essere indicati sempre preceduti dal simbolo dollaro ('**$**').

La scelta di rappresentare le variabili utilizzando il dollaro come prefisso è motivata dalla facilità con cui queste possono essere poi identificate durante la lettura del testo. Tuttavia, una scelta di questo tipo potrebbe essere discutibile, perché il dollaro non appartiene al nome della variabile e perché potrebbe indurre il lettore a utilizzarlo sempre, anche quando negli script non si deve. Quindi, il buon senso deve guidare nella decisione finale.

### 3.7.5 Comandi e istruzioni

A volte si ha la necessità di indicare un comando, o un'istruzione, all'interno del testo normale. Per questo, è opportuno utilizzare un carattere a spaziatura fissa, come nel caso dei nomi di file e directory, però qui si pone un problema nuovo dovuto alla possibile presenza di spazi e trattini. I programmi di composizione normali tendono a interrompere le righe, quando necessario, in corrispondenza degli spazi ed eventualmente anche dei trattini. Se il comando o l'istruzione che si scrive è breve, è consigliabile l'utilizzo di spazi e trattini non interrompibili.[8]

Quando si utilizza SGML (compreso HTML), si può usare l'entità '** **' per indicare uno spazio non interrompibile, mentre se si usa solo LaTeX, è il carattere tilde ('**~**') che ha questa funzione.

Il problema del trattino non è semplice, perché non esiste un trattino generico non separabile, fine a se stesso. Di trattini ne esistono di varie misure e non sempre esistono corrispondenti per diversi tipi di programmi di composizione.

[8] Naturalmente questo ha senso se poi il programma di composizione non tenta di suddividere le parole in sillabe.

### 3.7.6 Nomi di applicativi

Quando si fa riferimento al nome di un programma si pongono due alternative: l'indicazione del file eseguibile oppure del nome attribuito dall'autore al suo applicativo.

Per comprendere la differenza, si può pensare a Apache: il servente HTTP. Non si tratta di un semplice eseguibile, ma di un applicativo composto da diverse parti, in cui l'eseguibile è '**`httpd`**'. Nello stesso modo, nel caso di Perl (il linguaggio di programmazione), si può pensare all'applicativo in generale, composto dalle librerie e tutto ciò che serve al suo funzionamento; oppure si può voler fare riferimento solo all'eseguibile: '**`perl`**'.

I nomi di programmi applicativi dovrebbero essere indicati nello stesso modo in cui lo fa il loro autore, rispettando l'uso delle maiuscole e delle minuscole, in qualunque posizione del testo.

I nomi di questi tipi di entità non dovrebbero essere evidenziati in modo particolare.

### 3.7.7 Concetti e termini nuovi

I concetti e i termini che non si ritengono familiari per il lettore, dovrebbero essere evidenziati la prima volta che si presentano.

Per questo tipo di evidenziazione si utilizza un neretto oppure un corsivo. L'uso del neretto è contrario alla tradizione dei testi italiani, in cui questo viene fatto normalmente utilizzando solo il corsivo. Tuttavia, il neretto si presta meglio alla composizione in formati molto diversi; per esempio si ottiene facilmente anche su un documento da visualizzare attraverso uno schermo a caratteri.

### 3.7.8 Termini stranieri

A volte è opportuno utilizzare termini stranieri, non tradotti. Quando si tratta di termini non ben acquisiti nel linguaggio comune, almeno per il pubblico a cui si rivolge il documento, è opportuno utilizzare il corsivo tutte le volte in cui il termine viene adoperato.

Un termine tecnico può essere o meno di uso comune per il pubblico di lettori a cui si rivolge: se si tratta di un termine considerato normale per quell'ambiente, non è il caso di usare alcuna evidenziazione.

### 3.7.9 Nomi proprietari e logotipi

L'indicazione di nomi che fanno riferimento a marchi di fabbrica o simili, va fatta come appare nel copyright o nella nota che fa riferimento al brevetto, rispettando l'uso delle maiuscole e dell'eventuale punteggiatura. Si dovrebbe evitare, quindi, di prendere in considerazione un eventuale logo grafico del prodotto. Non è opportuno fare risaltare maggiormente i nomi di questo tipo.[9]

All'interno del testo non è conveniente fare riferimento al detentore del copyright o del brevetto. Eventualmente, di questo problema dovrebbero farsi carico delle note opportune all'inizio del documento che si scrive.[10]

[9] A questa regola si può aggiungere che, nel caso il nome sia scritto utilizzando solo lettere maiuscole, può essere opportuno limitarsi a indicarlo utilizzando solo l'iniziale maiuscola, lasciando il resto in minuscolo.

[10] In generale, non è indispensabile fare alcun tipo di riferimento di questo genere, se lo scopo di ciò che si scrive non è quello di trattare espressamente di questo o quel prodotto.

### 3.7.10 Titoli

Nei testi di lingua italiana, i titoli vanno scritti come se si trattasse di testo normale, con le particolarità seguenti:

- non viene mai posto il punto fermo finale;
- si cerca di evitare l'inserzione di altri segni di punteggiatura, a meno che ciò sia necessario per qualche motivo;
- non si usano evidenziazioni particolari di parole o nomi come invece potrebbe avvenire nel testo normale.

Un documento a carattere tecnico viene normalmente suddiviso in segmenti a più livelli. Per avere maggiore facilità nella trasformazione del documento in diversi formati tipografici finali, conviene limitare la scomposizione a un massimo di due livelli.

### 3.7.11 Didascalie

Gli elementi che non fanno parte del flusso normale di un documento, come tabelle e figure, sono accompagnate generalmente da un titolo e da una didascalia. Il titolo serve a identificarle, mentre la didascalia ne descrive il contenuto.

I titoli di tabelle, figure e oggetti simili, seguono le regole dei titoli normali, mentre il testo delle didascalie segue le regole del testo normale. Tuttavia, quando si utilizzano programmi di composizione che permettono di abbinare solo una nota descrittiva, che funga sia da titolo che da didascalia, occorre fare una scelta:

- quando le note sono brevi, è opportuno che si comportino come i titoli, cioè non contengano simboli di punteggiatura;
- quando sono più lunghe, si può decidere di trattarle come didascalie vere e proprie, con tutti i simboli di punteggiatura necessari per una comprensione corretta del contenuto.

Naturalmente, la scelta fatta deve valere per tutte le descrizioni che si abbinano a questi oggetti di un particolare documento: brevi o lunghe che siano.

### 3.7.12 Elenchi descrittivi

Gli elenchi descrittivi, come quelli che si ottengono con HTML utilizzando la struttura seguente, possono essere insidiosi, perché potrebbero tradursi in modo differente a seconda del tipo di programma di composizione utilizzato.

```
<dl>
<dt>Primo elemento</dt>
<dd>
    <p>Descrizione del primo elemento,...
    Bla bla bla...</p>
</dd>
</dl>
```

L'elemento descrittivo dell'elenco è in pratica un titolo che introduce una parte di testo generalmente rientrata. Sotto questo aspetto, la voce descrittiva segue le regole già viste per i titoli.

Tuttavia, il problema sta nel fatto che si potrebbe essere indotti a riprendere un discorso lasciato in sospeso quando veniva introdotto l'elenco, come nell'esempio seguente:

```
Bla bla bla bla...

Primo elemento

        Descrizione del primo elemento,...
        Bla bla bla...

Qui si riprende il discorso precedente all'elenco descrittivo.
...
```

Infatti, l'utilizzo dei rientri fa percepire immediatamente la conclusione dell'elenco stesso. Quando si scrive un documento che deve poter essere convertito in molti formati differenti, che quindi potrebbe essere elaborato da programmi di composizione di vario tipo, può darsi che i rientri vengano perduti e gli elementi descrittivi dell'elenco appaiano come dei titoli veri e propri. Ma se ciò accade, quando si ricomincia «il discorso lasciato in sospeso», sembra che questo appartenga all'argomento dell'ultimo titolo apparso.

```
Bla bla bla bla...

Primo elemento

Descrizione del primo elemento,...
Bla bla bla...

Qui si riprende il discorso precedente all'elenco descrittivo.
...
```

Pertanto, se si vogliono utilizzare strutture di questo tipo, è consigliabile che appaiano alla fine di una sezione, quando quello che viene dopo è un titolo di una sezione o di qualcosa di simile.

### 3.7.13 Richiami in nota

I richiami in nota (le note a piè pagina e quelle alla fine del documento) sono composti con le stesse regole del testo normale. Quando il riferimento a una nota si trova alla fine di una parola cui segue un segno di interpunzione, è opportuno collocare tale riferimento dopo il simbolo di interpunzione stesso.

### 3.7.14 Indicizzazione

La costruzione di un indice analitico deriva dall'inserzione di riferimenti all'interno del testo, attraverso istruzioni opportune definite dal tipo di programma usato per la composizione.

Le voci inserite in questi riferimenti, che poi formeranno l'indice analitico, vanno scelte in modo da essere uniformi, secondo alcune regole molto semplici.

- Si utilizzano le lettere minuscole, a meno che si tratti di nomi particolari che vanno sempre scritti in un modo prestabilito:
  - i nomi proprietari vanno scritti come indicato dalla casa produttrice;
  - i nomi di applicativi software vanno scritti come indicato dall'autore;
  - i nomi di file e directory vanno scritti esattamente come sono, tenendo conto che i file eseguibili vanno indicati senza percorso, mentre gli altri dovrebbero contenerlo;

  - i nomi di variabili di ambiente vanno scritti esattamente come sono, prefissati dal simbolo dollaro.

- Si utilizza solo il singolare;

I riferimenti per la generazione dell'indice analitico vanno posti preferibilmente nei luoghi opportuni, in modo da evitare inutili rimandi a pagine che non contengono ciò che si cerca. Per esempio, la parola **file** potrebbe trovarsi in quasi tutte le pagine di un testo di informatica, mentre è conveniente che l'indice analitico riporti solo le pagine in cui si parla del concetto che questa parola rappresenta.

I nomi di programmi eseguibili e di file di dati standard, come per esempio i file di configurazione, dovrebbero essere inseriti nell'indice analitico ogni volta che appaiono nel testo.

### 3.7.15 Riferimenti bibliografici e simili

Esiste una forma precisa e molto articolata per la stesura delle bibliografie, che corrisponde allo standard ISO 690. A ogni modo, vale la regola generale per cui un riferimento bibliografico deve contenere tutti i dati necessari a reperire il documento a cui si fa riferimento. In condizioni normali, le informazioni essenziali per identificare una pubblicazione sono quelle seguenti:

- l'autore o gli autori;
- il titolo completo;
- l'editore;
- la data di edizione;
- il numero ISBN (se disponibile);
- l'URI (se il documento è disponibile attraverso la rete).

Generalmente è consigliabile comporre gli elenchi bibliografici indicando le opere a partire dall'autore, mettendo il titolo in testo corsivo o inclinato, separando le varie componenti di ogni riferimento bibliografico attraverso delle virgole, come nell'esempio seguente:

> Claudio Beccari, *LaTeX, Guida a un sistema di editoria elettronica*, Hoepli, 1991, ISBN 88-203-1931-4

Se non si dispone di un sistema automatico per la gestione dei riferimenti bibliografici, quando si cita un documento all'interno del testo, è bene seguire alcune regole elementari.

- I riferimenti ad altri documenti, all'interno del testo normale, vanno fatti indicando il titolo completo, in corsivo o inclinato, aggiungendo il nome dell'autore o degli autori.
- Il titolo è separato con una virgola da un eventuale sottotitolo.
- I riferimenti a un testo già citato possono essere fatti utilizzando solo il titolo o solo l'autore, o attraverso altri mezzi, purché si sia certi di non creare ambiguità o disagio al lettore.

Segue un esempio molto semplice di come può essere fatto un riferimento del genere all'interno del testo normale:

... Questa sezione fa riferimento a concetti contenuti in *LaTeX, Guida a un sistema di editoria elettronica*, di Claudio Beccari...

## 3.8 Riferimenti


- Michele Dalla Silvestra, *Scrittura testi per l'ILDP*
- Robert Kiesling, *The LDP Style Mini-HOWTO*
  *<http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html>*
- Claudio Beccari, *LaTeX, Guida a un sistema di editoria elettronica*, Hoepli, 1991, ISBN 88-203-1931-4
- M. Fazio, *Dizionario e manuale delle unità di misura*, Zanichelli
- *Scienza, tecnologia e arte della stampa e della comunicazione*, Arti poligrafiche europee
  *<http://www.apenet.it/grafica/libri/Grafica/Grafica01/indice02.html>*
  *<http://www.apenet.it/grafica/libri/Grafica/Grafica02/indice02.html>*
  *<http://www.apenet.it/grafica/libri/Grafica/Grafica03/indice02.html>*
- *Scienza, tecnologia e arte della stampa e della comunicazione*: Giuseppe Orsello, *Preparazione del manoscritto*, Arti Poligrafiche Europee
  *<http://www.apenet.it/grafica/libri/Grafica/Grafica01/1206.html>*
- Marco Gaiarin, *Linux Italian HOWTO*
  *<http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html>*
- Maurizio Pistone, *Lingua italiana e altra linguistica*
  *<http://www.freeweb.org/letteratura/pistone/linguaitaliana.html>*
- *Dictionnaire panlatin de l'informatique*
  *<http://www.tele3.net/dicoinfo/_bdt.htm>*
- *NetGlos - The Multilingual Glossary of Internet Terminology*
  *<http://wwli.com/translation/netglos/netglos.html>*
- *Amiga Translators' Organization*
  *<http://bilbo.di.unipi.it/~ato-it/>*
- Bureau International des Poids et Mesures
  *<http://www.bipm.org/>*
- Bureau International des Poids et Mesures, *Le Système international d'unités (SI)*
  *<http://www.bipm.org/pdf/brochure-si.pdf>*
- Bureau International des Poids et Mesures, *The International System of Units (SI)* (traduzione in inglese)
  *<http://www.bipm.org/pdf/si-brochure.pdf>*
- National Institute of Standards and Technology, *International System of Units (SI)*
  *<http://physics.nist.gov/Pubs/SP330/sp330sl.pdf>*

- National Institute of Standards and Technology, *Guide for the Use of the International System of Units (SI)*, 1995

  *<http://physics.nist.gov/cuu/pdf/sp811.pdf>*

- Markus Kuhn, *Standardized Units for Use in Information Technology*, 1995

  *<http://www.cl.cam.ac.uk/~mgk25/information-units.txt>*

- National Institute of Standards and Technology, *Prefixes for binary multiples*

  *<http://physics.nist.gov/cuu/Units/binary.html>*

- *Excerpts from ISO 690-2, Information and documentation -- Bibliographic references -- Part 2: Electronic documents or parts thereof*

  *<http://www.nlc-bnc.ca/iso/tc46sc9/standard/690-2e.htm>*

# Evoluzione dell'editoria elettronica

Con il termine «editoria elettronica», si vuole fare riferimento agli strumenti utilizzabili per produrre documentazione di buona qualità dal punto di vista tipografico. L'approccio di un programma per l'editoria può essere fondamentalmente di due tipi:

- a formattazione visuale o WYSIWYG (*What you see is what you get*);[1]
- a composizione differita.

Nel primo caso, durante la stesura, il documento appare sullo schermo con lo stesso aspetto che avrebbe se venisse stampato in quel momento. Nel secondo, si scrive un file di testo normale con l'inserimento di comandi, come se si trattasse di un linguaggio di programmazione; quindi si passa alla composizione (una sorta di compilazione) attraverso la quale viene generato normalmente il file finale pronto per essere inviato alla stampa.

Il primo tipo di composizione è decisamente più pesante sotto l'aspetto elaborativo, prestandosi in particolare per i documenti brevi. Il secondo ha lo svantaggio di non permettere la verifica del risultato finale fino a quando non avviene la composizione, però richiede solo l'utilizzo di un programma normalissimo per la creazione e la modifica di file di testo, mentre solo al momento della composizione c'è bisogno di un'elaborazione consistente. In questo senso è più adatto alla redazione di documenti di grandi dimensioni.

Raramente si riescono a trovare programmi in grado di conciliare entrambe le esigenze. Nel sistema operativo Dos, il programma Ventura Publisher è stato un precursore di questa doppia filosofia: permetteva sia la formattazione visuale, sia quella differita, basandosi su un sorgente che poteva essere modificato con un programma di scrittura a caratteri.

## 4.1 Evoluzione

L'editoria elettronica non è più solo cartacea. In particolare esistono gli ipertesti, cioè documenti elettronici la cui consultazione avviene attraverso riferimenti e non in modo puramente sequenziale.

In questo senso, se l'editoria elettronica viene vista come mezzo di documentazione generale non più orientata a un supporto particolare, non può avere immediatamente una rappresentazione finale definitiva. Per esempio, un documento in HTML non potrà mai essere identico a un documento stampato.

Quando si vuole produrre un documento compatibile con diversi tipi di supporti (carta, ipertesto HTML, guida interna, ecc.) non si possono avere pretese stilistiche particolari; quindi, un programma visuale diventa quasi inutile.

A fianco di questi problemi di compatibilità, si aggiungono delle esigenze nuove, come per esempio la possibilità di estrarre dal documento elettronico determinati tipi di informazioni necessarie ad alimentare una base di dati. In questo senso, le informazioni cercate, oltre che riconoscibili all'interno del formato utilizzato, devono essere coerenti e complete.

Comunque, anche nell'ambito dell'editoria cartacea tradizionale, la prima esigenza che è stata sentita è quella dell'uniformità stilistica, cosa che sarebbe bene fosse controllabile anche attraverso il sistema elettronico di composizione.

[1] «Ciò che si vede è ciò che si ottiene»

## 4.2 Codifica del testo (markup)

Il termine *markup* (o marcatura) deriva dall'ambiente tipografico dove è stato usato per definire le annotazioni fatte su una bozza, allo scopo di segnalare al compositore o al dattilografo il modo con cui alcune parti del testo andavano evidenziate o corrette. A tale proposito, esiste uno standard nella simbologia da utilizzare in questi casi, che si può trovare ancora nei libri di tipografia. Queste annotazioni simboliche possono riferirsi all'aspetto dei caratteri, all'allineamento dei paragrafi, alle spaziature e via dicendo.

Nell'editoria elettronica, il concetto alla base del termine *markup* si è esteso in modo da includere i simboli speciali, o meglio, la codifica inserita nel testo per permetterne l'elaborazione.

Volendo generalizzare, la codifica del testo è tutto ciò che ne esplicita l'interpretazione. A livello umano, la stessa punteggiatura e certe forme di spaziatura, sono la codifica che serve a chiarire il significato del testo, diventando parte essenziale di questo. Oggi non sarebbe comprensibile separare concettualmente la punteggiatura dal testo, però in passato è stato così. Basta pensare ai telegrammi, o all'apparizione di questi simboli nella storia della scrittura.

### 4.2.1 Linguaggio di markup

La tecnica di composizione del testo utilizzando l'inserimento di marcatori o di codici, richiede la definizione di una serie di convenzioni, tali da definire un ***linguaggio di markup***. Un tale linguaggio deve specificare quale tipo di marcatura è utilizzabile, quale è richiesta, in che modo si distingua dal testo e quale sia il suo significato.

I linguaggi di *markup* possono essere diversi e si distinguono due gruppi fondamentali: linguaggi procedurali e linguaggi descrittivi.

Un linguaggio di *markup* procedurale serve a definire il processo da svolgere in un punto particolare del documento. È come un linguaggio di programmazione in cui si usano chiamate di funzioni, o di procedure, per compiere le operazioni richieste. Per esempio può trattarsi di ordini riferiti alla scrittura del testo, allo spostamento, alla definizione di margini, del salto pagina e di tutto ciò che si rende necessario. In questo senso, un linguaggio di *markup* procedurale consente generalmente la definizione completa di tutto ciò che serve a stabilire l'aspetto finale del documento stampato (o visualizzato).

Un linguaggio di *markup* descrittivo, al contrario, usa la codifica dei marcatori per classificare le parti del documento, dando loro un nome. In pratica, si delimitano queste porzioni di testo e si definisce la loro appartenenza a una categoria determinata, identificata da un nome. In tal modo, questo tipo di linguaggio di *markup* non è in grado di fornire indicazioni sull'aspetto finale del documento, in quanto il suo scopo è solo quello di definire la struttura del testo. Evidentemente sarà compito di un'altra applicazione utilizzare le informazioni sulla struttura del testo per generare un formato finale, secondo regole e definizioni stabilite al di fuori del linguaggio descrittivo stesso.

### 4.2.2 Vantaggi di un linguaggio descrittivo

Un linguaggio di *markup* descrittivo, nel momento in cui non si prende carico di definire l'aspetto finale del documento, pone l'accento sul contenuto e non sull'apparenza. Questo è fondamentale quando il «documento» viene inteso come informazione pura che possa materializzarsi in forme molto diverse.

L'informazione «pura», in quanto tale, richiede anche che sia espressa attraverso un formato indipendente dalle piattaforme, ma soprattutto che sia indipendente dai formati proprietari.

## 4.3 SGML

SGML è un linguaggio di *markup* descrittivo, definito dallo standard *ISO 8879: Information processing---Text and office systems---Standard Generalized Markup Language (SGML)*, 1986. L'SGML è uno standard internazionale per la definizione di metodi di rappresentazione del testo in forma elettronica in modo indipendente dall'hardware e dal sistema utilizzato.

### 4.3.1 Linguaggio descrittivo

Come accennato, l'SGML è un linguaggio di *markup* descrittivo. Questo permette a un documento steso secondo questo linguaggio, di essere elaborato da programmi differenti, per scopi diversi, dove la stampa o comunque la semplice lettura testuale del contenuto sia solo uno dei tanti possibili obiettivi da raggiungere. Si è già accennato alla possibilità di estrarre informazioni da un documento per l'utilizzo in una base di dati e questo particolare dovrebbe essere sufficiente per intuire il senso di tale approccio descrittivo.

### 4.3.2 Definizione del tipo di documento

Il linguaggio SGML utilizza il concetto di «tipo di documento» e di «definizione del tipo di documento». Per la precisione di parla di DTD, ovvero, *Document type definition*. In pratica, nell'ambito dell'SGML, è necessario che sia stato definito il modo in cui i vari elementi del testo possono essere utilizzati. Ciò che non è definito, non può essere usato, ma quello che è stato definito deve rispettare le regole stabilite.

A titolo di esempio, si può immaginare la definizione di un tipo di documento riferito alla scrittura di lettere commerciali. La lettera deve contenere degli elementi essenziali: il mittente, uno o più destinatari, la data, l'oggetto, il corpo, l'indicazione di colui che la firma e la sigla del dattilografo che la scrive materialmente. Tutti questi elementi devono essere presenti, probabilmente anche con un certo ordine (l'indicazione di chi firma deve trovarsi in fondo e non all'inizio). Inoltre, questi elementi possono scomporsi in altri elementi più dettagliati; per esempio, l'informazione sulla persona che firma può comporsi della qualifica, il titolo personale, il nome e il cognome. Il DTD deve prendersi carico di definire tutto questo, stabilendo ciò che è valido e cosa invece non lo è.

In questo modo, poi, un documento SGML può essere analizzato da un programma speciale, l'analizzatore SGML (*SGML parser*), per la verifica del rispetto di queste regole, prima di utilizzare in qualunque modo questo documento.

L'SGML, assieme al DTD, garantendo l'uniformità dei documenti dello stesso tipo, consente di uniformare i procedimenti successivi. Per tornare all'esempio precedente, da un punto di vista di puro contenuto del testo, non dovrebbe essere importante l'ordine degli elementi che lo compongono, quando sia possibile distinguerli. Tuttavia, una lettera che inizia con la firma e finisce con l'indicazione del destinatario, non è scritta nel modo corretto; così il DTD potrebbe essere progettato in modo da imporre un certo ordine, a vantaggio delle elaborazioni successive.

### 4.3.3 Indipendenza dei dati

Nella definizione di SGML si è affermato che si tratta di uno standard indipendente dall'hardware e dal sistema utilizzato. Questa indipendenza riguarda la rappresentazione del testo, che non può fare affidamento su una codifica particolare.

Si pensi all'uso di lettere accentate e di simboli speciali che non possono essere rappresentati con lo standard tradizionale dell'ASCII a 7 bit. Si pensi a cosa accadrebbe se un testo scritto con caratteri ISO Latin 1 venisse elaborato in un sistema configurato per una codifica differente: quei simboli e quelle lettere potrebbero risultare modificati. D'altro canto, la stessa scrittura di determinati caratteri potrebbe essere un problema, non disponendo di una tastiera adatta.

Ecco quindi il significato dell'indipendenza dall'hardware (fondamentalmente la tastiera) e dal sistema (principalmente la codifica dei simboli utilizzati).

Per ottenere questo risultato, l'SGML utilizza un meccanismo di sostituzione di stringhe, attraverso quelle che vengono chiamate ***entità***, per mezzo del quale si stabilisce il rimpiazzo di tali entità con qualcosa di adeguato, quando il documento viene elaborato.

# URI

Un URI (*Uniform resource identifier*) è un indirizzo espresso attraverso una stringa di caratteri per identificare una risorsa fisica o astratta. La risorsa in questione è un'entità e la sua collocazione non si trova necessariamente all'interno di una rete. In pratica, il concetto di URI incorpora i concetti di URL (*Uniform resource locator*) e di URN (*Uniform resource name*).

Un URL identifica una risorsa rappresentando il metodo di accesso a questa; un URN identifica la risorsa attraverso un nome, che deve essere unico a livello globale e deve persistere anche quando la risorsa cessa di esistere o diventa inaccessibile.

## 5.1 Trascrivibilità

L'esigenza primaria degli indirizzi URI è la loro «trascrivibilità». Con questo termine si vuole fare riferimento alla facilità con la quale questi devono poter essere trascritti, sia a livello meccanico, sia a livello umano. In pratica:

- un URI è composto da una sequenza di «caratteri» e non necessariamente da ottetti (byte);
- un URI deve poter essere trascritto attraverso qualunque mezzo, come una pubblicazione stampata o un appunto fatto a mano, in tal senso non può utilizzare caratteri particolari che possono mancare in un contesto determinato;
- un URI deve poter essere ricordato facilmente dalle persone, per cui è utile che la stringa che rappresenta un URI abbia un significato che ne faciliti la memorizzazione.

Dal momento che ci deve essere la possibilità di rappresentare un URI all'interno di parentesi di qualsiasi tipo, i caratteri corrispondenti a queste parentesi non possono essere utilizzati letteralmente all'interno di un indirizzo del genere. Le parentesi in questione sono quelle tonde, quadre, graffe e angolari: '`(`', '`)`', '`[`', '`]`', '`{`', '`}`', '`<`', '`>`'.

## 5.2 Sintassi

La sintassi di un URI è piuttosto complessa, perché dipende molto dal contesto a cui si applica. Non è il caso si entrare troppo nel dettaglio; piuttosto è meglio apprendere la logica della cosa.

***schema*** **:** ***parte_successiva_dipendente_dallo_schema***

Quello che si vede è il modello di prima approssimazione di un indirizzo URI assoluto (verrà trattato in seguito il concetto di URI relativo). In questa prima fase si distinguono due parti, separate da due punti verticali ('`:`'), dove prima appare un nome che definisce uno «schema» e poi continua con una stringa che va interpretata in base alle regole specifiche di quello schema.

La sintassi di un URI non stabilisce a priori quale sia la forma che deve avere la stringa che segue i due punti; tuttavia, è frequente l'utilizzo di URI secondo i modelli seguenti:

***schema*** **:** **//** ***autorità*** **[** ***percorso*** **[** **?** ***interrogazione*** **]** **]**

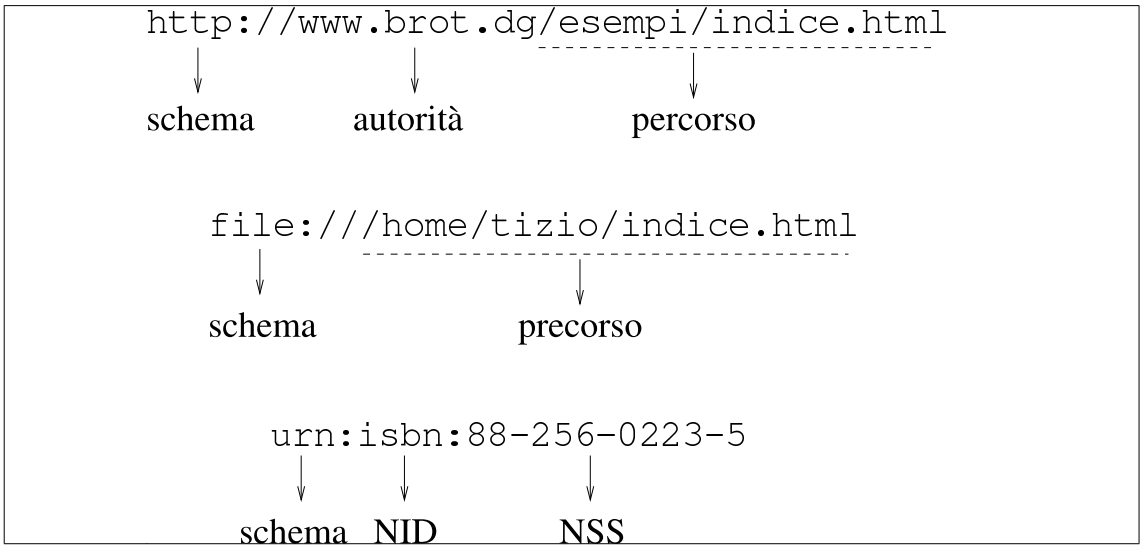
***schema*** **:** **/** ***percorso***

Convenzionalmente, quando una risorsa viene individuata attraverso un URI che per sua natura contiene un'informazione gerarchica, la separazione tra i vari livelli di questa gerarchia avviene utilizzando una barra obliqua normale ('`/`'). Si tratta evidentemente di una tecnica ereditata dal

file system Unix; tuttavia, ciò resta indipendente dal fatto che la risorsa in questione risieda fisicamente all'interno di un file system o meno.

La figura 5.1 mostra alcuni esempi a proposito di URI composti secondo i modelli più frequenti.

Figura 5.1. Esempi di URI comuni.




Nella figura si vede anche un caso particolare, riferito a un URN di tipo ISBN (*International standard book number*). Lo schema di un URN è sempre '**`urn:`**'; a questo segue l'indicazione di un NID (*Namespace identifier*), ovvero un identificatore che qualifica l'informazione successiva; infine si inserisce l'informazione, definita NSS (*Namespace specific string*), ovvero ciò che va inteso nel contesto stabilito dal NID. L'esempio che appare nella figura fa riferimento al numero ISBN 88-256-0223-5, esprimendolo in forma di URN.

### 5.2.1 Accesso a un servente attraverso la rete

Quando l'indirizzo URI si riferisce a un servizio offerto attraverso la rete, la struttura di ciò che è stato definito come «autorità» si articola in modo particolare:

[ *utente* [ : *parola_d'ordine* ] @ ] *nodo* [ : *porta* ]

In questo modo si può specificare il nominativo utente per l'accesso alla risorsa, eventualmente anche la parola d'ordine (benché ciò sia decisamente sconsigliabile per motivi di sicurezza), quindi il nodo che offre il servizio e infine la porta del servizio.

Il nodo può essere indicato per nome, attraverso il nome di dominio, oppure attraverso il numero IPv4. Purtroppo non è stato definito un modo per indicare un numero IPv6, dal momento che la sua forma renderebbe impossibile l'interpretazione corretta dell'indirizzo.

Se si omettono le informazioni riferite all'utente, vuol dire che queste non sono necessarie, oppure che esistono dei valori predefiniti per questo; per quanto riguarda la porta del servizio, se questa non viene indicata si fa riferimento sempre al suo valore predefinito. Naturalmente, è stabilito dal servente quali siano i valori predefiniti.

### 5.2.2 Riferimento agli URI

Per sua natura, l'indirizzo URI è un riferimento a una risorsa. In generale vanno considerate anche due circostanze particolari: il riferimento a un frammento della risorsa e l'indicazione di URI relativi.

Un URI relativo è un indirizzo ridotto che parte da un punto di partenza conosciuto. Il principio deriva dal concetto di percorso relativo all'interno di un file system. In generale, un URI relativo può essere indicato omettendo tutta la parte iniziale che si possa determinare altrimenti.

Di fronte a un URI che contenga un'informazione sul percorso in forma gerarchica, è abbastanza facile intendere cosa sia la base di riferimento per gli URI relativi: basta togliere dall'indirizzo attuale tutto quello che segue l'ultima barra obliqua. Per esempio, per il documento *`http://www.brot.dg/esempi/articolo.html`* l'URI di base è *`http://www.brot.dg/esempi/`*, per cui, il riferimento a '`figure/foto.jpg`' richiama effettivamente l'URI *`http://www.brot.dg/esempi/figure/foto.jpg`*.

Il percorso di un URI relativo può essere indicato anche con una barra obliqua iniziale, ma in questo caso si farà riferimento a un percorso assoluto nell'ambito dell'URI. Continuando con l'esempio precedente, il riferimento a '`/nuovo/documento.html`' richiama effettivamente l'URI *`http://www.brot.dg/nuovo/documento.html`*.

In presenza di un percorso relativo, è possibile utilizzare anche i simboli '`.`' e '`..`', con lo stesso significato che hanno nel file system Unix: il primo rappresenta la posizione corrente e il secondo quella precedente.

È importante osservare che il riferimento alla stringa nulla indica implicitamente lo stesso URI iniziale.

> Il problema degli URI relativi non è così semplice come è stato descritto. In realtà vanno prese in considerazione altre cose, come per esempio la possibilità che il tipo di risorsa (di solito in un documento HTML) possa incorporare l'informazione esplicita di un URI di base.

Quando il tipo di risorsa lo consente, è possibile aggiungere all'URI l'indicazione di un frammento particolare. Questa parte aggiuntiva la si riconosce perché è preceduta dal simbolo '`#`':

```
http://www.brot.dg/esempi/articolo.html#commento
```

L'esempio mostra il riferimento al frammento '**`#commento`**' nell'ambito dell'URI '`http://www.brot.dg/esempi/articolo.html`'. Dal momento che la stringa nulla fa riferimento alla risorsa attuale, i riferimenti interni alla stessa risorsa sono indicati facilmente attraverso il solo frammento:

```
#commento
```

L'esempio mostra un riferimento relativo al frammento '**`#commento`**' della risorsa corrente.

### 5.2.3 Esempi

Frequentemente, il nome dello schema dell'indirizzo URI corrisponde al nome del protocollo necessario per raggiungere la risorsa relativa. I più comuni sono:

- '**`http`**'
- '**`ftp`**'
- '**`gopher`**'
- '**`mailto`**'
- '**`wais`**'
- '**`telnet`**'
- '**`tn3270`**'
- '**`news`**'

Quando si vuole fare riferimento a un file locale senza utilizzare alcun protocollo particolare, si può indicare anche lo schema '**`file`**', ma in questo caso ci sono delle particolarità che verranno mostrate dagli esempi.

- `http://www.brot.dg:8080/esempi/indice.html`
  - protocollo HTTP
  - nodo *`www.brot.dg`*
  - porta 8080
    Viene indicata la porta perché si vuole fare riferimento a un valore diverso dallo standard che per il protocollo HTTP è 80
  - risorsa '`/esempi/indice.html`'
- `http://www.brot.dg/esempi/indice.html`
  Come nell'esempio precedente, ma senza l'indicazione della porta che questa volta corrisponderà al valore predefinito, cioè 80.
- `http://192.168.1.1/esempi/indice.html`
  Come nell'esempio precedente, ma l'indicazione del nodo avviene per mezzo del suo indirizzo IPv4 invece che attraverso il nome di dominio.
- `ftp://ftp.brot.dg/pub/archivi/esempio.tar.gz`
  - protocollo FTP
  - nodo *`ftp.brot.dg`*
  - risorsa '`/pub/archivi/esempio.tar.gz`'
- `ftp://tizio@ftp.brot.dg/pub/archivi/esempio.tar.gz`
  Come nell'esempio precedente, con la differenza che si fa riferimento a un utente particolare.

- `ftp://tizio:segretissima@ftp.brot.dg/pub/archivi/esempio.tar.gz`

  Come nell'esempio precedente, con la differenza che si aggiunge l'indicazione della parola d'ordine di accesso al servizio, cosa che in generale è bene non passare mai in questo modo.

- `file://localhost/home/daniele/indice.html`

  In questo caso si vuole fare riferimento a un file locale. Precisamente si tratta del file '`/home/daniele/indice.html`' contenuto nell'elaboratore *`localhost`*.

  Questo tipo di indicazione è utile specialmente quando si vuole fare riferimento a una pagina indice o iniziale, caricata automaticamente all'atto dell'avvio di un programma cliente per la navigazione.

- `file:///home/daniele/indice.html`

  Esattamente come nell'esempio precedente, con la differenza che si omette l'indicazione esplicita dell'elaboratore locale: *`localhost`*.

- `file:/home/daniele/indice.html`

  Esattamente come nell'esempio precedente, con la differenza che si utilizza una sola barra obliqua dopo l'indicazione '**`file:`**' (ma in generale è preferibile la forma precedente, con le tre barre oblique).

- `mailto:tizio@dinkel.brot.dg`

  Si tratta di un indirizzo di posta elettronica, nel quale è essenziale fornire l'indicazione del nominativo utente. Dopo il nome del nodo di destinazione non appare un percorso, perché in questo caso non avrebbe significato.

## 5.3 Limitazioni nell'uso dei caratteri

Ogni componente di un URI ha delle regole proprie nell'uso dei caratteri, dal momento che alcuni di questi hanno significati speciali. Purtroppo le regole in questione sono tante e la cosa migliore che si può fare è quella di usare il buon senso, riservando la lettura della documentazione specifica ai casi in cui è indispensabile chiarire il problema nel dettaglio (RFC 2396).

In generale non è ammissibile l'uso dello spazio. Infatti, considerato il principio di trascrivibilità degli URI, lo spazio dovrebbe essere inteso solo come una necessità legata al tipo di trascrizione utilizzata. Per il resto, se la propria lingua lo consente, sarebbe bene limitarsi all'uso delle lettere dell'alfabeto latino (maiuscole e minuscole, ma senza accenti), le cifre numeriche e alcuni simboli: '**`@`**', '**`*`**', '_', '**`-`**' e il punto ('**`.`**'). Gli altri simboli possono creare problemi di trascrivibilità o avere significati particolari (basta pensare alle barre oblique e ai due punti verticali).

Quando un simbolo particolare non può essere utilizzato in modo letterale nel contesto in cui lo si vuole inserire, può essere indicato attraverso una notazione speciale: '***`%hh`***'. La sigla ***`hh`*** rappresenta una coppia di cifre esadecimali. A questa regola fa eccezione lo spazio che viene codificato normalmente con il segno '**`+`**', ma non in tutte le occasioni (di solito solo nelle stringhe di richiesta).

Generalmente, per gli indirizzi URI normali non c'è la necessità di preoccuparsi di questo problema, anche la tilde può essere utilizzata letteralmente nell'indicazione dei percorsi. La tabella 5.1 mostra l'elenco di alcune corrispondenze tra simboli particolari e la codifica alternativa utilizzabile negli URI.

Tabella 5.1. Alcune corrispondenze tra simboli particolari e codifica alternativa utilizzabile negli URI.

| Carattere | Codifica corrispondente |
|---|---|
| % | %25 |
| & | %26 |
| + | %2B |
| / | %2F |
| = | %3D |

In linea di principio, un URI dovrebbe essere realizzato in modo da non dover utilizzare questa tecnica di protezione per i caratteri «speciali». La situazione più probabile in cui è necessario utilizzare questo procedimento è riferito alle stringhe di interrogazione.

## 5.4 Verifica degli URI con Checkbot

Checkbot [1] è un programma Perl molto semplice da utilizzare, per controllare la validità degli indirizzi contenuti in una pagina HTML locale o remota. Il suo utilizzo è molto semplice e il rapporto che si ottiene è molto dettagliato, consentendo una comprensione chiara del tipo di errore che impedisce di raggiungere qualche indirizzo URI. Tutto viene gestito attraverso un eseguibile unico denominato '**`checkbot`**':

`checkbot` [***opzioni***] [***uri_iniziale***...]

Nella situazione più semplice, si utilizza Checkbot specificando un solo indirizzo URI iniziale da scandire: se si tratta di una pagina HTML, vengono analizzati tutti i riferimenti contenuti al suo interno. Per esempio così:

```
$ checkbot file:///home/tizio/prova.html
```

Come si vede, è opportuno indicare sempre il riferimento alla pagina da scandire utilizzando un URI, anche se si tratta di un file locale.

Leggendo la pagina di manuale *checkbot(1)*, si possono trovare tante opzioni per questo programma. Tuttavia, il suo funzionamento normale non richiede nulla, salvo forse la necessità di indicare un proxy, quando questo è indispensabile per raggiungere la rete esterna (con l'opzione '**`--proxy`** ***uri***').

Se non si indica nulla di diverso attraverso le opzioni della riga di comando, la scansione genera il file '`checkbot.html`' e un altro file il cui nome rispetta il modello '`checkbot-`***nodo***`.html`'. Il primo di questi due è un riepilogo dell'esito della scansione, mentre il secondo elenca dettagliatamente gli URI per i quali c'è stato qualche problema. Comunque, si raggiunge il secondo attraverso un riferimento ipertestuale presente nel primo.

## 5.5 Riferimenti


- T. Berners-Lee, R. Fielding, U.C. Irvine, L. Masinter, *RFC 2396: Uniform Resource Identifiers (URI): General Syntax*, 1998

  *<http://www.cis.ohio-state.edu/cgi-bin/rfc/rf2396.html>*

  *<http://www.cis.ohio-state.edu/cs/Services/rfc/rfc-text/rfc2396.txt>*


[1] **Checkbot** stesse condizioni di Perl

- International ISBN agency, *The ISBN Users' Manual*
  *<http://www.isbn.org/standards/home/isbn/International/ISBNmanual.asp>*

# SGML: introduzione

L'SGML non è un «linguaggio di scrittura» da imparare e usare così com'è. Al contrario, è un linguaggio per definire il modo in cui il testo deve essere scritto: solo dopo si può iniziare a scrivere secondo le regole stabilite.

Volendo fare un abbinamento con i linguaggi di programmazione, sarebbe come se prima si dovesse definire il linguaggio stesso, per poi poter scrivere i programmi secondo quelle regole.

La descrizione fatta in questo capitolo potrebbe risultare noiosa, considerato che solo dopo molte sezioni si mostra in che modo realizzare effettivamente il proprio DTD e applicarlo a un documento. Considerata la complessità dei concetti espressi, si ritiene più conveniente una spiegazione che parte dal basso, piuttosto che usare un approccio inverso, che presumerebbe una conoscenza minima di partenza.

## 6.1 DTD: definizione del tipo di documento

Le regole che definiscono la struttura e la scomposizione del documento, assieme a quasi tutte le altre che governano la logica dell'SGML, sono contenute nel DTD.

Queste regole possono essere permissive o restrittive, in funzione degli obiettivi che ci si prefigge; ovvero, in funzione del contenuto di quel tipo di documento e delle cose che con questo ci si aspetta di fare.

La complessità del mondo reale, fa sì che non ci sia modo di realizzare un DTD unico che vada bene per tutti gli scopi. Un DTD ipotetico, che volesse andare bene un po' per tutto, dovrebbe essere anche qualcosa di estremamente generico e permissivo, annullando tutti i benefici dell'utilizzo dell'SGML.

Esempi reali di DTD «tuttofare» sono quelli delle prime versioni dell'HTML, in cui tutto si concentra nella definizione di elementi il cui scopo prevalente è definire, anche se solo vagamente, l'aspetto finale che dovrebbe avere il risultato. Lo scopo dell'SGML non è quello di stabilire il risultato finale del documento, tuttavia, si può benissimo predisporre un DTD orientato a questo obiettivo. Ma questo, nel caso dell'HTML, giustifica poi l'estrema debolezza della sua struttura, dove è ammesso quasi tutto.

È difficile comprendere subito il significato pratico di questo approccio: la definizione del tipo di documento e poi la scrittura del testo. Lo si può comprendere solo quando si lavora assiduamente nell'ambito della produzione di documentazione, quando ci si accorge che le proprie esigenze sono diverse da quelle degli altri, per cui diventa difficile adattarsi all'uso di modelli già esistenti.

## 6.2 Elementi

Dal punto di vista di SGML, una singola unità di testo la cui dimensione varia a seconda del contesto, è un ***elemento***, a cui si impone l'attribuzione di un nome. SGML non fornisce alcun modo per attribuire un significato agli elementi del testo, tranne per il fatto di avergli dato un nome. Piuttosto, attraverso un analizzatore SGML, è possibile verificare che questi siano collocati correttamente secondo le relazioni stabilite.

I nomi degli elementi, sono definiti tecnicamente ***identificatori generici***, utilizzando la sigla GI (*Generic identifier*).

Nel sorgente SGML, gli elementi sono indicati normalmente attraverso l'uso di marcatori che hanno la forma consueta '**<**...**>**' e '**</**...**>**', dove il primo inizia l'elemento nominato tra le parentesi

angolari e il secondo chiude l'elemento. Per esempio, si potrebbe definire l'elemento '**acronimo**' e utilizzarlo nel testo nel modo seguente:

```
...Il gruppo <acronimo>ILDP</acronimo> si occupa di...
```

Il significato che questo elemento può avere, non è definito dall'SGML. Il fatto di avere delimitato l'elemento '**acronimo**' potrebbe servire per estrarre dal documento tutte le sigle utilizzate, per inserire queste in un indice particolare, oppure solo per fini stilistici di evidenziamento uniforme.

La difficoltà nella scrittura di un testo in SGML si riduce a questo: utilizzare i marcatori necessari a identificare correttamente i vari elementi del testo, secondo le regole stabilite nella definizione del documento stesso (il DTD).

### 6.2.1 Abbreviazioni nell'indicazione degli elementi

Prima ancora di iniziare a vedere il contenuto del DTD, è bene chiarire che esistono altri modi per delimitare un elemento SGML. Per la precisione, si tratta di abbreviazioni di cui alcuni autori non riescono a fare a meno. La scrittura di un sorgente SGML è un po' come quella di un sorgente di un linguaggio di programmazione: si può essere concisi o prolissi. Di solito, quando si è concisi si scrive del codice difficile da leggere, mentre in generale è meglio scrivere tutto in forma chiara senza risparmiare. L'esempio visto in precedenza,

```
...Il gruppo <acronimo>ILDP</acronimo> si occupa di...
```

può essere abbreviato in

```
...Il gruppo <acronimo>ILDP</> si occupa di...
```

e anche nel modo seguente, che però porta con sé un vincolo importante: non si possono usare delle barre oblique all'interno dell'elemento abbreviato in questo modo.

```
...Il gruppo <acronimo/ILDP/ si occupa di...
```

Con questi sistemi, oltre a rendere il sorgente SGML poco leggibile, si rischia di non ottenere i risultati che si attendono se gli strumenti di elaborazione utilizzati non riconoscono tali estensioni del linguaggio.

### 6.2.2 Primo impatto con un DTD

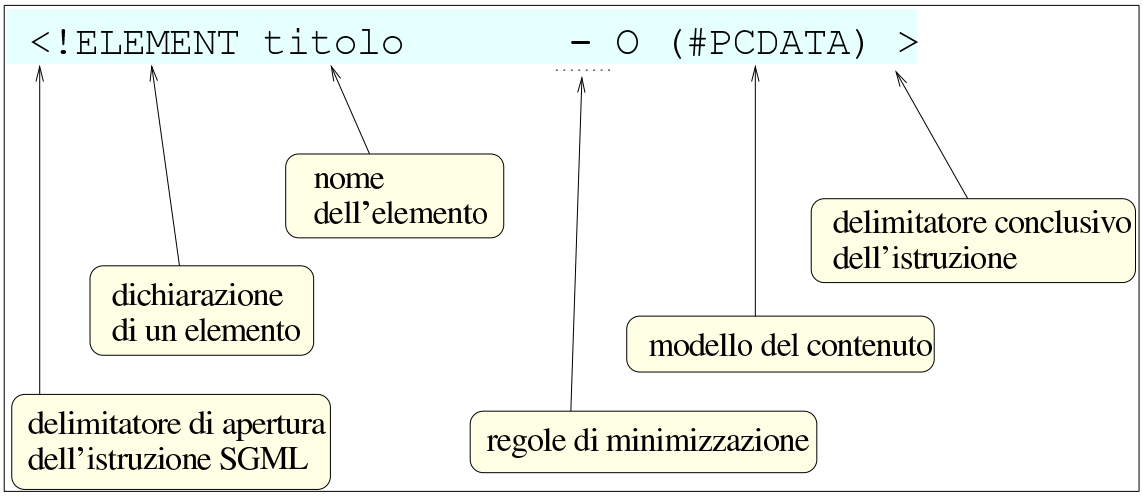
La definizione del DTD è ottenuta da una serie di istruzioni dichiarative composte secondo una sintassi molto semplice. L'esempio seguente rappresenta le istruzioni necessarie a definire gli elementi di un tipo di documento ipotetico definibile come '**relazione**'.

```
<!ELEMENT relazione     - - (titolo?, data, contenuto) >
<!ELEMENT titolo        - o (#PCDATA) >
<!ELEMENT data          - o (#PCDATA) >
<!ELEMENT contenuto     - o (paragrafo+, firma+) >
<!ELEMENT paragrafo     - o (#PCDATA) >
<!ELEMENT firma         - o (#PCDATA) >
```

Ognuna delle righe che appaiono nell'esempio rappresenta una dichiarazione di un elemento SGML. Una dichiarazione, di qualunque tipo, è delimitata da una parentesi angolare aperta (il simbolo di minore), seguita immediatamente da un punto esclamativo ('**<!**') e da una parentesi angolare chiusa ('**>**').

La dichiarazione di un elemento si compone poi della parola chiave '**ELEMENT**', seguita dal nome dell'elemento, dalle regole di minimizzazione rappresentate da due caratteri e da un modello del contenuto.

Figura 6.1. Scomposizione delle varie parti della dichiarazione di un elemento SGML.




Le varie parti che compongono qualunque tipo di dichiarazione SGML sono separate da spazi orizzontali (caratteri spazio, o tabulazioni orizzontali) oppure anche da interruzioni di riga, permettendo così di proseguire le istruzioni su più righe distinte.

## 6.3 Regole di minimizzazione

Le ***regole di minimizzazione***, rappresentate da due caratteri staccati, indicano l'obbligatorietà o meno dell'utilizzo del marcatore di apertura e di chiusura per l'elemento dichiarato. Il primo dei due simboli rappresenta l'apertura, il secondo la chiusura. Un trattino indica che il marcatore è obbligatorio, mentre la lettera '**O**' sta per «opzionale» e indica così che può essere omesso:

| | |
|---|---|
| `- -` | sono obbligatori entrambi i marcatori; |
| `- O` | è obbligatorio il marcatore iniziale, mentre quello finale è facoltativo; |
| `O -` | il marcatore iniziale è facoltativo, mentre quello finale è obbligatorio (di solito non capita questa situazione); |
| `O O` | sono facoltativi entrambi i marcatori. |

Nell'esempio mostrato in precedenza, solo l'elemento '**relazione**' richiede l'utilizzo di marcatori di apertura e di chiusura, mentre tutti gli altri possono essere indicati utilizzano il solo marcatore di apertura. In pratica, il contesto permette di individuare dove finiscano tali elementi nel testo.

La possibilità o meno di rendere facoltativo l'uso dei marcatori di apertura e di chiusura non è solo un fatto di gusto, in quanto dipende anche dall'organizzazione del tipo di documento. Se le dichiarazioni diventano ambigue, non si possono più distinguere gli elementi nel testo SGML.

## 6.4 Modello del contenuto

La parte finale della dichiarazione di un elemento SGML è il ***modello del contenuto***, che si distingue perché è racchiuso tra parentesi tonde. Serve a descrivere il tipo di contenuto che può avere l'elemento e si può esprimere attraverso parole riservate che hanno un significato preciso, come '**#PCDATA**' (*Parsed character data*) che rappresenta una qualunque sequenza di caratteri valida, oppure attraverso l'indicazione di nomi di altri elementi che possono (o devono) essere contenuti in qualche modo.

Il modello del contenuto, può articolarsi in modo molto complesso, allo scopo di definire le relazioni tra gli elementi contenuti.

Per il momento, è bene osservare che un elemento, il cui modello del contenuto sia composto esclusivamente della parola riservata '**#PCDATA**', non può contenere al suo interno altri tipi di elementi. Il significato di alcune delle parole riservate più comuni, utilizzabili per definire il contenuto di un elemento, sono riportate più avanti in questo capitolo, dopo la presentazione di altri concetti essenziali, necessari per comprenderne il senso.

### 6.4.1 Indicatori di ripetizione

Il modello del contenuto utilizza un sistema abbastanza complesso per definire la possibilità di contenere più elementi dello stesso tipo e per indicare raggruppamenti di elementi. Per indicare la ripetizione, viene usato un simbolo alla fine dell'oggetto a cui si riferisce, chiamato ***indicatore di ripetizione*** (*occurrence indicator*):

| | |
|---|---|
| `+` | il segno '**+**' usato come suffisso, rappresenta una o più ripetizioni dell'elemento; |
| `?` | il segno '**?**' usato come suffisso, rappresenta zero o al massimo un'occorrenza dell'elemento; |
| `*` | il segno '*****' usato come suffisso, rappresenta zero o più ripetizioni dell'elemento; |
| | se non viene usato nessun suffisso, l'elemento indicato deve essere usato esattamente una volta. |

Dall'esempio mostrato in precedenza viene ripreso l'estratto seguente, nel quale si può osservare che: l'elemento '**titolo**' può apparire al massimo una volta all'interno di '**relazione**' (precisamente all'inizio di questo elemento); l'elemento '**paragrafo**' deve essere contenuto almeno una volta all'interno dell'elemento '**contenuto**' (lo stesso vale per l'elemento '**firma**'); l'elemento '**firma**' può contenere solo caratteri normali senza altri elementi.

```
<!ELEMENT relazione     - - (titolo?, data, contenuto) >
<!ELEMENT contenuto     - O (paragrafo+, firma+) >
<!ELEMENT firma         - O (#PCDATA) >
```

### 6.4.2 Connettori

Quando un elemento deve poter contenere diversi tipi di elementi, è necessario usare dei simboli, detti ***connettori***, per stabilirne la relazione:

| | |
|---|---|
| `,` | la virgola ('**,**') indica che l'elemento precedente e quello successivo devono apparire nell'ordine in cui sono; |
| `&` | la e-commerciale ('**&**') indica che l'elemento precedente e quello successivo devono essere presenti entrambi, ma possono apparire in qualunque ordine; |

| | |
|---|---|
| \| | la barra verticale (‘**|**’) indica che solo uno tra i due elementi che connette può apparire. |

Riprendendo il solito estratto dell’esempio già mostrato precedentemente, si può osservare l’uso della virgola in qualità di connettore:

```
<!ELEMENT relazione     - - (titolo?, data, contenuto) >
<!ELEMENT contenuto     - O (paragrafo+, firma+) >
<!ELEMENT firma         - O (#PCDATA) >
```

L’elemento ‘**relazione**’ può contenere al massimo un titolo all’inizio, quindi deve apparire un elemento ‘**data**’ e dopo di questo anche un elemento ‘**contenuto**’. L’elemento ‘**contenuto**’ deve contenere uno o più elementi ‘**paragrafo**’ a partire dall’inizio, mentre in coda deve avere uno o più elementi ‘**firma**’.

```
<!ELEMENT nominativo    - - (nome & cognome) >
<!ELEMENT voce          - - (punto | numero) >
```

Per completare gli esempi sull’uso dei connettori, si osservi quanto mostrato sopra. L’elemento ‘**nominativo**’ deve contenere un elemento ‘**nome**’ e un elemento ‘**cognome**’, in qualunque ordine; l’elemento ‘**voce**’ può contenere solo un elemento a scelta tra ‘**punto**’ e ‘**numero**’.

### 6.4.3 Raggruppamenti

All’interno di un modello di contenuto, è possibile indicare dei raggruppamenti che esprimono in pratica dei sottomodelli, a cui poter applicare gli indicatori di ripetizione e i connettori. Per questo si usano le parentesi tonde. Si osservi l’esempio seguente:

```
<!ELEMENT figure - - ( (eps | ph), img*, caption?) >
```

L’elemento ‘**figure**’ deve contenere un’occorrenza del sottogruppo ‘**(eps | ph)**’, zero o più ripetizioni dell’elemento ‘**img**’ e al massimo un’occorrenza di ‘**caption**’, nell’ordine descritto. Il sottogruppo ‘**(eps | ph)**’ rappresenta una singola occorrenza di ‘**eps**’ oppure ‘**ph**’.

Quando si utilizzano gli operatori di ripetizione assieme ai raggruppamenti, possono nascere degli equivoci. Ammesso che ciò possa avere senso, si osservi la variante seguente dell’esempio già presentato:

```
<!ELEMENT figure - - ( (eps | ph)+, img*, caption?) >
```

È stato aggiunto il segno ‘**+**’ dopo il gruppo ‘**(eps | ph)**’. In questo modo, si intende che sia possibile l’inserimento iniziale di una serie indefinita di elementi ‘**eps**’ o ‘**ph**’, in qualunque ordine, purché ce ne sia almeno uno dei due. Quindi, non è necessario che si tratti solo di elementi ‘**eps**’ o solo di ‘**ph**’.

### 6.4.4 Eccezione

Se nella definizione di un elemento si vogliono indicare delle eccezioni a quanto definito dal modello di contenuto, si può indicare un gruppo di elementi successivo al modello del contenuto.

Questo gruppo può essere preceduto dal segno ‘**+**’ o dal segno ‘**-**’ indicando rispettivamente un’eccezione di inclusione, o un’eccezione di esclusione.

```
<!ELEMENT address - O (#PCDATA) +(newline) >
```

L'esempio mostra l'elemento '**address**' contiene caratteri normali, ma che può includere eccezionalmente anche l'elemento '**newline**'.

```
<!ELEMENT acronimo - - (#PCDATA) -(acronimo) >
```

L'esempio mostra l''elemento '**acronimo**' contiene caratteri normali e che non può includere se stesso (a essere precisi, non è necessario dichiarare una cosa del genere, dal momento che il contenuto '**#PCDATA**' non ammette altri elementi al suo interno).[1]

### 6.4.5 Elementi vuoti

Alcuni tipi di elementi non sono fatti per circoscrivere una zona di testo, ma solo per rappresentare qualcosa che si trova in un certo punto. Questi elementi, non vengono dichiarati con un modello di contenuto tra parentesi, ma con l'utilizzo della parola chiave '**empty**'.

L'esempio seguente, dichiara l'elemento '**toc**' che non può contenere alcunché.

```
<!ELEMENT toc - O EMPTY>
```

Tipicamente, tali elementi, sono dichiarati in modo che il marcatore di chiusura sia solo facoltativo. Non potendo contenere alcunché, sarebbe perfettamente inutile renderlo obbligatorio.

## 6.5 Dichiarazione multipla

Eventualmente, un gruppo di elementi che abbiano le stesse caratteristiche, cioè le stesse regole di minimizzazione e lo stesso modello del contenuto, può essere dichiarato in una sola istruzione. L'esempio seguente dovrebbe essere sufficiente a comprendere il meccanismo.

```
<!ELEMENT ( annotazione | avvertimento | pericolo ) - - (#PCDATA) >
```

## 6.6 Attributi

Un elemento può prevedere la presenza di uno o più attributi. Si tratta di informazioni che non compongono il contenuto dell'elemento, ma di qualcosa che, non potendo apparire nel testo, serve per qualche ragione ai programmi che elaborano successivamente il documento. Il classico esempio è costituito da quei marcatori utilizzati per i riferimenti incrociati. L'esempio seguente mostra l'uso di un elemento vuoto, denominato '**ref**', contenente l'attributo '**point**' a cui viene dato il valore '**esempio**':

```
Si veda il capitolo <ref point="esempio"> che contiene
molti esempi utili al riguardo.
```

È importante osservare che il valore assegnato a un attributo deve essere delimitato attraverso apici doppi (come mostrato nell'esempio), oppure attraverso apici singoli. Eccezionalmente, è possibile assegnare un valore senza alcuna delimitazione, quando si tratta di una sola parola composta da lettere alfabetiche, cifre numeriche, trattino normale ('**-**'), trattino basso ('_'), due punti ('**:**').

L'esempio seguente mostra la dichiarazione dell'elemento '**ref**', già presentato nell'esempio, tenendo conto che il suo scopo è quello di essere utilizzato come riferimento a una parte del documento identificata attraverso il valore assegnato all'attributo '**point**'.

[1] In questo momento può apparire strano l'uso di questa forma di eccezione. Tuttavia, per comprenderne meglio il senso, occorrerebbe conoscere come funzionano le entità parametriche che sono descritte più avanti. Con queste si può definire un modello del contenuto attraverso una sorta di variabile e, in tal caso, potrebbe essere conveniente l'indicazione di una o più eccezioni, sia in aggiunta che in detrazione.

```
<!ELEMENT ref - O EMPTY>
<!ATTLIST ref
        point IDREF #REQUIRED
        name CDATA "riferimento">
```

Attraverso l'istruzione '**ATTLIST**' si definiscono gli attributi di un elemento. Dopo l'indicazione del nome dell'elemento a cui si fa riferimento, segue l'elenco degli attributi, ognuno dei quali inizia con un codice di interruzione di riga seguito eventualmente da altri tipi di spazi. Ciò significa che l'istruzione '**ATTLIST**' deve essere composta proprio come indicato dall'esempio, solo i rientri sono facoltativi.

L'esempio indica che l'elemento '**ref**' contiene due attributi: '**point**' e '**name**'. Il primo è obbligatorio ('**#REQUIRED**'), mentre per il secondo è stato indicato un valore predefinito, nel caso non venga specificato espressamente ('**riferimento**').

Il tipo di contenuto di un attributo viene definito attraverso delle parole chiave, che possono essere indicate usando lettere maiuscole o minuscole indifferentemente. Di seguito ne vengono descritte alcune:

| | |
|---|---|
| CDATA | rappresenta una stringa di qualunque tipo di carattere, ammettendo anche simboli di punteggiatura o altro, che comunque mantiene solo il suo significato letterale (*Character data*); |
| NMTOKEN | rappresenta qualunque tipo di carattere alfanumerico (lettere, numeri e spazi soltanto), che dovrebbe comporre un nome (*Name token*); |
| NUMBER | rappresenta solo cifre numeriche, cioè un numero; |
| ID | rappresenta un identificatore unico per quel tipo di documento, costituito da un nome senza caratteri speciali o segni di punteggiatura, che viene utilizzato successivamente per farvi riferimento; |
| IDREF | indica che l'attributo deve essere un puntatore valido a un identificatore di un attributo '**ID**', corrispondente in un altro elemento. |

È importante osservare che la parole chiave '**CDATA**' viene usata anche in altre situazioni con un significato simile, ma non identico. Nel caso definisca il contenuto di un attributo, è ammesso l'uso di macro (entità) che vengono espanse.

Il tipo di contenuto di un attributo, può essere indicato in modo preciso attraverso una serie di scelte alternative. In tal caso, invece di utilizzare le parole chiave già elencate, si indicano le stringhe alternative, separate dalla barra verticale, tra parentesi tonde. Per esempio, '**(bozza | finale)**' rappresenta la possibile scelta tra le due parole '**bozza**' e '**finale**'.

L'ultimo dato da inserire per ogni attributo è il valore predefinito, oppure una parola chiave a scelta tra le seguenti:

| | |
|---|---|
| #REQUIRED | rappresenta l'obbligatorietà dell'inserimento del valore; |
| #IMPLIED | rappresenta un attributo facoltativo; |
| #CURRENT | in mancanza di un'indicazione esplicita, rappresenta l'utilizzo dell'ultimo valore assegnato allo stesso attributo dello stesso elemento. |

Tra tutti, merita attenzione la coppia '**ID**' e '**IDREF**'. Questi tipi di attributi possono essere molto utili per definire dei riferimenti incrociati all'interno del documento, quando la loro validità deve essere controllata con gli strumenti di convalida SGML. Si osservi l'esempio seguente:

```
<!ELEMENT label - O EMPTY>
<!ATTLIST label
        identity ID #REQUIRED>

<!ELEMENT ref - O EMPTY>
<!ATTLIST ref
        point IDREF #REQUIRED>
```

Nell'esempio si mostra la dichiarazione di un elemento '**label**' che non può contenere testo, in quanto serve solo per definire l'attributo '**identity**', di tipo '**ID**'. Questo permetterà l'utilizzo di marcatori simili a '**<label identity="miaetichetta">**', dove viene assegnato all'attributo '**identity**' un nome sempre diverso, allo scopo di identificare qualcosa. Sotto, la dichiarazione dell'elemento '**ref**' mostra un altro elemento che non può contenere testo, ma solo un attributo denominato '**point**', di tipo '**IDREF**', che può quindi contenere solo il nome di un identificatore già usato in un altro elemento con l'attributo '**ID**'.

In pratica, se nel testo SGML si dovesse utilizzare da qualche parte il marcatore '**<label identity="miaetichetta">**', in un altro punto sarebbe valido il marcatore '**<ref point="miaetichetta">**', perché l'identificatore '**miaetichetta**' esiste effettivamente.

Ricapitolando, un attributo '**ID**' di un marcatore è valido quando è unico nel documento SGML che si scrive, mentre un attributo '**IDREF**' è valido quando esiste il valore corrispondente di un attributo '**ID**'.

Spesso, per cose del genere, si preferisce usare attributi di tipo '**CDATA**', per permettere l'utilizzo di caratteri di ogni tipo, togliendo però all'SGML la possibilità di controllare la validità di tali riferimenti incrociati.

## 6.7 Entità

Con questo termine, ***entità***, si fa riferimento a due tipi di oggetti: macro per la sostituzione di stringhe (entità generali) o macro per la sostituzione di nomi all'interno di istruzioni SGML (entità parametriche).

Le macro per la sostituzione di stringhe, una volta dichiarate, si utilizzano all'interno del sorgente SGML come abbreviazioni o come un modo per identificare lettere o simboli che non possono essere usati altrimenti. Per esempio, utilizzando le entità ISO 8879:1986, la frase

```
Wer bekommt das größte Stück Torte?
```

può essere scritta nel sorgente nel modo seguente:

```
Wer bekommt das gr&ouml;&szlig;te St&uuml;ck Torte?
```

Le entità generali, quindi, sono identificate nel testo SGML perché iniziano con la e-commerciale ('**&**') e terminano con un punto e virgola. È bene osservare che il punto e virgola non è obbligatorio in ogni situazione, ma solo quando il carattere successivo sia diverso da uno spazio orizzontale o da un codice di interruzione di riga. In generale, però, sarebbe bene usare sempre il punto e virgola. La tabella 6.6 elenca alcune macro delle entità standard più importanti.

Tabella 6.6. Alcune macro delle entità standard secondo le specifiche ISO 8879:1986.

| | | | |
|---|---|---|---|
| á | á | Á | Á |
| â | â | Â | Â |
| à | à | À | À |
| å | å | Å | Å |
| ã | ã | Ã | Ã |
| ä | ä | Ä | Ä |
| æ | æ | Æ | Æ |
| ç | ç | Ç | Ç |
| é | é | É | É |
| ê | ê | Ê | Ê |
| è | è | È | È |
| ë | ë | Ë | Ë |
| í | í | Í | Í |
| î | î | Î | Î |
| ì | ì | Ì | Ì |
| ï | ï | Ï | Ï |
| ñ | ñ | Ñ | Ñ |
| ó | ó | Ó | Ó |
| ô | ô | Ô | Ô |
| ò | ò | Ò | Ò |
| ø | ø | Ø | Ø |
| õ | õ | Õ | Õ |
| ö | ö | Ö | Ö |
| ß | ß | | |
| ú | ú | Ú | Ú |
| û | û | Û | Û |
| ù | ù | Ù | Ù |
| ü | ü | Ü | Ü |
| ý | ý | Ý | Ý |
| ÿ | ÿ | | |
| & | & | @ | @ |
| * | * | | |
| ˆ | ^ | ˜ | ~ |
| © | © | | |
| $ | $ | % | % |
| # | # | | |
| ! | ! | ¡ | ¡ |
| ? | ? | ¿ | ¿ |
| ‐ | - | _ | _ |
| \ | \ | | |
| " | " | | |
| < | < | > | > |
| [ | [ | ] | ] |
| { | { | } | } |

Le entità standard ISO 8879, sono distinte in 19 gruppi, che in parte si sovrappongono (a volte si ripetono alcune dichiarazioni nello stesso modo). Questi 19 gruppi di entità corrispondono ad altrettanti file, per i quali esiste anche un nome stabilito.

L'altro tipo di macro, riguarda invece la sostituzione all'interno delle istruzioni SGML, cioè nella

dichiarazione del DTD.

L'esempio seguente mostra la dichiarazione dell'elemento '**p**' che può contenere l'elemento o gli elementi indicati all'interno della macro '**%inline;**'.

```
<!ELEMENT p - O (%inline;) >
```

La dichiarazione di un'entità avviene utilizzando l'istruzione '**ENTITY**'. L'esempio seguente mostra la dichiarazione di un'entità da utilizzare nel sorgente SGML.

```
<!ENTITY agrave "\'a">
```

In questo caso, si vuole che la macro '**à**' venga sostituita con la stringa '**\'a**'. Evidentemente, questa trasformazione non ha niente a che vedere con SGML. È semplicemente una scelta motivata dal tipo di programma utilizzato successivamente per rielaborare il risultato generato dall'analizzatore SGML.

L'esempio seguente mostra la dichiarazione di due entità da utilizzare all'interno delle istruzioni SGML.

```
<!ENTITY % emph " em | concept | cparam ">
<!ENTITY % inline "(#PCDATA | %emph;)*">
```

La dichiarazione di questo tipo di entità si distingue perché viene utilizzato il simbolo di percentuale subito dopo la parola '**ENTITY**', staccandolo da questa e anche dal nome dell'entità successivo. Anche in questo caso si utilizza solo come pura sostituzione di stringhe, per cui la dichiarazione di '**%inline;**', facendo a sua volta riferimento a '**%emph;**', è equivalente a quella seguente:

```
<!ENTITY % inline "(#PCDATA | em | concept | cparam )*">
```

Naturalmente, una macro può contenere anche il riferimento a un'altra macro. Per esempio, la dichiarazione dell'ipotetico elemento '**p**', fatta nel modo seguente,

```
<!ELEMENT p - O (%inline;) >
```

è equivalente alla dichiarazione:

```
<!ELEMENT p - O ((#PCDATA | em | concept | cparam )*) >
```

### 6.7.1 Acquisizione dall'esterno

Le entità di qualunque tipo, possono essere dichiarate abbinando una stringa a una macro, come è stato mostrato in precedenza. In alternativa, a una macro si può abbinare un file esterno (file inteso nel senso più ampio possibile). In tal caso, si utilizza la parola chiave '**SYSTEM**' come nell'esempio seguente:

```
<!ENTITY capitolo2 SYSTEM "capitolo2.sgml">
```

In tal modo, quando nel documento SGML si utilizza la macro '**&capitolo2;**' e poi lo si elabora attraverso un analizzatore SGML, si ottiene l'inserimento del file '`capitolo2.sgml`'. Più o meno ciò che si fa normalmente con le direttive di un preprocessore di un linguaggio di programmazione.

Nello stesso modo si può fare per dichiarare un'entità parametrica, come nell'esempio seguente:

```
<!ENTITY % isoent SYSTEM "isoent.txt">
```

L'esempio mostra la dichiarazione della macro '**`%isoent;`**', riferita al file '`isoent.txt`'. Per utilizzare questa macro, bisogna sapere a cosa si riferisce; trattandosi di un file, è logico pensare che si tratti di un testo articolato su più righe, quindi inadatto all'inserzione all'interno delle istruzioni. Generalmente, una macro del genere serve a incorporare un pezzo di DTD dall'esterno.

```
%isoent;
```

Come si vede dall'esempio, è normale vedere la chiamata di una macro di questo tipo, da sola, all'esterno di qualunque istruzione del DTD. L'esempio mostrato è comunque significativo: rappresenta l'inclusione di un file che presumibilmente, dal nome, serve a incorporare le entità ISO, cioè quelle standard riferite alle lettere accentate e ai simboli speciali.

A questo proposito, potrebbero esistere diversi file, del tipo: '`isoent.latex.txt`', '`isoent.html.txt`',... che prima di avviare l'analizzatore SGML vengono sostituiti al file '`isoent.txt`', in modo da ottenere la sostituzione corretta in base all'elaborazione successiva che si vuole ottenere (LaTeX, HTML, ecc.).

Se non fosse ancora chiaro, ecco come potrebbe essere composto l'ipotetico file '`isoent.txt`' quando si vogliono le sostituzioni corrette per LaTeX.

```
<!ENTITY agrave "\‘a">
<!ENTITY Agrave "\‘A">
<!ENTITY egrave "\‘e">
<!ENTITY Egrave "\‘E">
<!ENTITY eacute "\’e">
<!ENTITY Eacute "\’E">
...
```

L'acquisizione di una macro da un file esterno può essere dichiarata senza specificare esplicitamente il file, lasciando che l'analizzatore trovi il file corretto in base a un catalogo SGML. L'argomento verrà ripreso in seguito, comunque, in questo tipo di dichiarazione, manca il nome del file.

```
<!ENTITY capitolo2 SYSTEM>
```

```
<!ENTITY % isoent SYSTEM>
```

Solitamente, si preferisce includere in questo modo solo le macro parametriche, cosa che potrà essere compresa intuitivamente in seguito.

### 6.7.2 Codici macro speciali

È bene ribadire che l'uso delle entità standard (ISO), permette di rendere il testo SGML indipendente dalla piattaforma utilizzata. Tuttavia, la dichiarazione della sostituzione dipende dalla piattaforma e, come si è mostrato, si tendono a predisporre diversi schemi di sostituzione per le diverse piattaforme a cui si vuole fare riferimento.

In situazioni eccezionali, può essere conveniente indicare i caratteri per numero, decimale o esadecimale, attraverso una notazione simile a quella delle entità normali. Per esempio, se si usa la codifica ISO 8859-1 (Latin 1), la macro '**`è`**', oppure la macro '**`è`**' corrisponderà alla lettera '**è**' (la «e» accentata normale).

Questa possibilità è fondamentale proprio quando si definiscono le stringhe di sostituzione per una piattaforma determinata (hardware-software), in cui si debbano indicare caratteri speciali identificati dal numero corrispondente.

```
<!ENTITY egrave "&#232;">
```

Potrebbe sembrare che un testo SGML non possa utilizzare una codifica particolare, quale ISO 8859-1 o altro. Non è così. L'SGML mette a disposizione le entità standard, ma ciò non toglie che si possa decidere di usare comunque una codifica (ASCII) estesa come Latin 1 o altro. Ovviamente questo rende il testo dipendente dalla piattaforma, precisamente dalla codifica. Volendo essere precisi, la codifica utilizzabile dipende dalla dichiarazione SGML.

## 6.8 Sezioni marcate

Le sezioni marcate sono una specialità di SGML, poco usata e poco conosciuta. Si tratta di istruzioni che vengono inserite nel testo SGML (non nel DTD) e servono a vario titolo per delimitare del testo per qualche scopo.

Una sezione marcata si compone di un sorta di marcatore di apertura e di una sorta di marcatore di chiusura. Il marcatore di apertura contiene una parola chiave che ne identifica il comportamento. Si osservi l'esempio seguente:

```
<![INCLUDE[
Questa parte del testo è inclusa nell'elaborazione SGML.
]]>
```

Come si può intuire, la sezione marcata dell'esempio è introdotta da '**<![INCLUDE[**' ed è terminata da '**]]>**'. In questo caso, la parola chiave '**INCLUDE**' indica che il testo contenuto nella sezione marcata deve essere incluso nell'elaborazione (anche se ciò, per ora, può sembrare perfettamente senza significato).

Le parole chiave utilizzabili per definire la sezione marcata sono diverse; di seguito ne appare l'elenco.

| | |
|---|---|
| `INCLUDE` | Il contenuto della sezione marcata deve essere incluso nel documento SGML e deve essere elaborato normalmente. |
| `IGNORE` | Il contenuto della sezione marcata deve essere escluso dal documento SGML. Se l'analizzatore SGML genera un qualche tipo di output, questo non conterrà tale sezione. |
| `CDATA` | Il contenuto della sezione marcata deve essere incluso e trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore o un'entità. Ciò vale per tutto, tranne il simbolo di chiusura della sezione marcata ('**]]>**'), che quindi è l'unica cosa che non può essere rappresentata all'interno di questa. |
| `RCDATA` | Il contenuto della sezione marcata deve essere incluso e trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore, ma continuando a espandere le entità. |
| `TEMP` | Il contenuto della sezione marcata deve essere inteso come «temporaneo». Ciò serve solo come riferimento umano, per localizzare facilmente una parte del documento che richiede una revisione o che dovrà essere rimossa. |

L'utilizzo di sezioni marcate di tipo '**INCLUDE**' e '**IGNORE**' è utile solo in abbinamento a entità parametriche. Prima di proseguire, è bene chiarire che quella specie di marcatore che apre una sezione marcata è come un'istruzione SGML, di quelle che appaiono nel DTD, anche se viene

usata al di fuori di questo, nel documento. In questo senso, al suo interno si possono usare le entità parametriche; quindi, una di queste macro può servire per definire in modo dinamico la parola chiave '**INCLUDE**' oppure '**IGNORE**', per decidere di includere o escludere quel blocco (e probabilmente anche altri) con la modifica di una sola macro.

Per esempio, nel DTD del documento potrebbe apparire la dichiarazione di un'entità parametrica denominata '**commentato**'.

```
<!ENTITY % commentato "INCLUDE">
```

Nel documento SGML potrebbero esserci una serie di sezioni marcate la cui inclusione deve dipendere da questa macro.

```
...
1 + 2 = 3
<![%commentato;[
La matematica non è un'opinione.
]]>
...
```

Quando il testo viene analizzato, la macro viene espansa e trovando che corrisponde a '**INCLUDE**', il testo delle sezioni marcate che l'hanno usata, vengono incluse. Al contrario, basta modificare la macro, assegnandole il valore '**IGNORE**', per fare in modo che tutte quelle sezioni marcate vengano ignorate.

Questo tipo di approccio potrebbe sembrare ugualmente scomodo per l'utilizzatore che non vuole toccare il DTD. Però, si vedrà in seguito che si possono inserire delle eccezioni al DTD nel preambolo di un documento SGML. Oppure, si può benissimo progettare un DTD con una componente esterna, destinata a questo tipo di ritocchi.

## 6.9 Dettagli importanti

Prima di passare alla descrizione dell'abbinamento di un DTD a un testo SGML, è bene chiarire alcuni dettagli che sono stati trascurati fino a questo punto.

### 6.9.1 Commenti

All'interno del documento sorgente SGML, come nel DTD, possono essere indicate delle righe di commento da non considerare parte del documento o della codifica. Queste si ottengono con i delimitatori '**<!--**' e '**-->**'.

Volendo approfondire meglio il problema, la sequenza '**<!>**' rappresenta l'istruzione SGML nulla e può essere usata indifferentemente nel DTD o nel sorgente SGML. In qualità di istruzione nulla viene ignorata semplicemente.

All'interno delle istruzioni SGML è possibile inserire dei commenti, attraverso una sequenza di due trattini ('**--**'), per aprire e chiudere il commento. Per esempio,

```
<!ELEMENT itemize - - (item+) -- elenchi puntati -- >
```

dichiara l'elemento '**itemize**' con un commento incorporato.

Questo dovrebbe chiarire il senso del commento composto da '**<!--**' e '**-->**': si tratta di un'istruzione (nulla) che contiene un commento.

Questa particolarità di SGML ha delle conseguente: nel testo che compone il commento, non possono apparire sequenze di due o più trattini.

### 6.9.2 Maiuscole e minuscole

Per convenzione, i nomi di entità sono sensibili alla differenza tra lettere maiuscole e minuscole, per cui '**À**' e '**à**' rappresentano rispettivamente la lettera «A» maiuscola con accento grave e la «a» minuscola con accento grave.

Per convenzione, i nomi degli elementi, i simboli delle regole di minimizzazione, i nomi degli attributi e le parole chiave, non sono sensibili alla differenza tra lettere maiuscole e minuscole. Quindi, nelle dichiarazioni del DTD,

```
<!element ref - o empty>
<!attlist ref
        id cdata #required
        name cdata "riferimento">
```

è identico a

```
<!ELEMENT REF - O EMPTY>
<!ATTLIST REF
        ID CDATA #REQUIRED
        NAME CDATA "riferimento">
```

così come nel testo SGML

```
... <ref id="capitolo-introduttivo" name="Intro"> ...
```

è identico a

```
... <REF id="capitolo-introduttivo" name="Intro"> ...
```

indifferentemente dal modo (maiuscolo o minuscolo) in cui l'elemento '**ref**' è stato dichiarato nel DTD.

Evidentemente, in generale, il contenuto delle stringhe delimitate è sensibile alla differenza tra maiuscole e minuscole, dal momento che riguarda i programmi che fanno uso del documento dopo l'analisi SGML; in pratica, dipende da questi programmi successivi il senso che hanno tali informazioni.

### 6.9.3 Delimitatori di stringa

In varie situazioni, all'interno del DTD e all'interno dei marcatori utilizzati nel testo SGML, può essere necessaria l'indicazione di stringhe. I simboli utilizzati per delimitare le stringhe possono essere gli apici doppi ('**"**…**"**'), oppure gli apici singoli ('**'**…**'**'). La scelta tra i due tipi di delimitatori dovrebbe essere indifferente, a parte la possibile necessità di inserire nelle stringhe proprio questi caratteri. Si osservi l'esempio seguente, in cui vengono dichiarate le entità riferite ad alcune lettere accentate da usare con LaTeX.

```
<!ENTITY uuml   '\"u'>
<!ENTITY Uuml   '\"U'>
<!ENTITY yacute "\'y">
<!ENTITY Yacute "\'Y">
```

In situazioni più complesse, potrebbe essere necessario indicare i caratteri con l'aiuto delle macro '**&#*nnn*;**', che permettono di identificare l'oggetto attraverso il numero corrispondente riferito al tipo di codifica utilizzato (purché il contesto preveda la successiva ulteriore espansione di tali macro).

### 6.9.4 Tipo di contenuto di un'entità generale

In precedenza, quando è stato mostrato in che modo possa essere definita un'entità, si è trascurato il fatto che si deve definire in che modo la stringa di sostituzione vada interpretata. Per questo, si aggiunge una parola chiave prima della stringa.

Se non si usa alcuna parola chiave, si intende che la stringa vada interpretata come appare, espandendo eventuali entità contenute al suo interno. Si osservi l'esempio.

```
<!ENTITY attenzione "&lt;ATTENZIONE&gt;">
```

Quando dovesse essere utilizzata la macro '**&attenzione;**', si otterrebbe la stringa '**<ATTENZIONE>**', perché le entità '**<**' e '**>**' vengono espanse ulteriormente.

Se si indica la parola chiave '**CDATA**', si intende che la stringa di sostituzione deve essere utilizzata in modo letterale, senza espandere alcuna sequenza che potrebbe sembrare un'entità.

```
<!ENTITY attenzione CDATA "&lt;ATTENZIONE&gt;">
```

L'esempio, modificato con l'introduzione della parola chiave '**CDATA**', fa sì che la macro '**&attenzione;**' si traduca in pratica in '**<ATTENZIONE>**', perché le entità '**<**' e '**>**' non vengono riconosciute come tali e quindi non vengono espanse.

Se si utilizza la parola chiave '**SDATA**' (*Special data*), si intende che la stringa di sostituzione deve essere utilizzata in modo letterale, senza espandere alcuna sequenza che potrebbe sembrare un'entità. Però, a differenza di '**CDATA**', l'informazione viene filtrata in modo particolare quando l'analizzatore SGML genera un risultato transitorio da riutilizzare con un altro programma di composizione.

### 6.9.5 Contenuto elementare degli elementi

In precedenza è già stata spiegata la dichiarazione degli elementi e la dichiarazione del contenuto. In particolare si è visto che attraverso la parola chiave '**#PCDATA**' si fa riferimento a testo normale che viene elaborato normalmente (*parsed*). Ciò significa che questo tipo di testo è soggetto alla sostituzione delle entità, come fino a questo punto si è dato per scontato.

Tuttavia esistono altre parole chiave per definire tipi di testo differenti. Segue l'elenco di quelle più comuni.

| | |
|---|---|
| `#PCDATA` | *Parsed character data*. Si riferisce a testo normale soggetto alla sostituzione delle entità. Questo testo non può contenere altri elementi. |
| `CDATA` | Il contenuto dell'elemento deve essere trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore o un'entità.<br>In pratica, la definizione di elementi con contenuto '**CDATA**' è decisamente sconsigliabile. Se esiste la necessità di delimitare una zona di testo da trattare in modo letterale, solitamente, si preferisce utilizzare una sezione marcata del tipo '**<![CDATA[…]]>**'. |
| `RCDATA` | Il contenuto dell'elemento deve essere trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore, ma continuando a espandere le entità. |

## 6.10 Abbinare il DTD al documento SGML

L'abbinamento di un DTD a un documento SGML avviene generalmente in modo formale. In presenza di situazioni eccezionali, questo abbinamento può essere implicito, come nel caso dell'HTML, ma è bene utilizzare ugualmente l'approccio generale anche in questi casi estremi.

Un sorgente SGML inizia normalmente con la dichiarazione del tipo di DTD utilizzato. Può trattarsi di un file esterno o di dichiarazioni incorporate nel documento stesso. Per esempio, la dichiarazione seguente indica all'analizzatore SGML di utilizzare un DTD esterno, denominato '**`linuxdoc`**' e contenuto nel file '`linuxdoc.dtd`'.

```
<!DOCTYPE linuxdoc SYSTEM "linuxdoc.dtd">
```

L'esempio seguente mostra invece una dichiarazione iniziale che contiene le istruzioni che compongono il DTD, racchiuse tra parentesi quadre.

```
<!DOCTYPE personale [
...
-- istruzioni SGML --
...
...
]>
```

Una terza possibilità permette di definire un file esterno e di aggiungere altre istruzioni particolari riferite al documento, come nell'esempio seguente, sempre utilizzando le parentesi quadre.

```
<!DOCTYPE linuxdoc SYSTEM "linuxdoc.dtd" [
...
-- istruzioni SGML --
...
...
]>
```

Inoltre, come è stato visto nel caso delle entità, l'acquisizione dall'esterno di un file contenente un DTD, può avvenire anche senza stabilire espressamente il nome di un file, lasciando che questo venga determinato da un catalogo. Così, l'esempio già visto del DTD '**`linuxdoc`**' si potrebbe trasformare nel modo seguente:

```
<!DOCTYPE linuxdoc SYSTEM>
```

Esiste anche un'altra alternativa: quella di indicare un identificatore pubblico, anch'esso riferito a un catalogo. Quello che segue è il preambolo di un file SGML scritto secondo il DTD HTML 3.2. Si osservi, a questo proposito, l'uso della parola chiave '**`PUBLIC`**'.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

### 6.10.1 Strategie

La scelta di incorporare il DTD nel documento, o di lasciarlo all'esterno, dipende da delle preferenze organizzative. Di sicuro, può essere sensato l'inclusione del DTD nel documento SGML quando si tratta di un DTD specifico che non viene usato altrove.

Nella realtà, si utilizzerà quasi sempre un DTD esterno, probabilmente predisposto da altri, abbinato a una serie di strumenti che permettono di produrre dei documenti in formato finale a partire dai sorgenti SGML scritti seguendo quel DTD particolare.

L'estensibilità del DTD resta sempre una possibilità utile per poter aggiungere delle entità interne o delle entità parametriche allo scopo di gestire opportunamente le sezioni marcate.

### 6.10.2 Entità interne

Come si è visto nella sezione precedente, la dichiarazione del DTD può includere delle istruzioni del DTD, generalmente estendendolo. Questo meccanismo permette, tra le altre cose, di inserire le cosiddette ***entità interne*** (*internal entity*). Si osservi l'esempio.

```
<!DOCTYPE linuxdoc SYSTEM
[
<!ENTITY pericolo "!!">
<!ENTITY posix    "POSIX">
<!ENTITY unix     "Unix">
<!ENTITY xwin     "X Window System">
<!ENTITY edizione "1999.12.31">
]>
```

L'esempio appena mostrato permette di utilizzare le macro '**`&pericolo;`**', '**`&posix;`**', '**`&unix;`**', '**`&xwin;`**' e '**`&edizione;`**', all'interno del sorgente SGML, ottenendo la loro sostituzione automatica. Per intenderne l'utilità, basta pensare al caso della macro '**`&xwin;`**' dell'esempio precedente: non occorre più ricordare come si deve scrivere («X», «X Window» o «X Window System»); se si decidesse di cambiare, basterebbe modificare la dichiarazione dell'entità. Il concetto è analogo a quello delle macro del preprocessore nei linguaggi di programmazione.

La definizione di entità interne è consentita anche quando queste dovessero essere già state dichiarate nel DTD. Le entità dichiarate nelle istruzioni aggiuntive, dovrebbero prendere la precedenza e sostituirsi a quelle eventualmente già dichiarate nel DTD.

### 6.10.3 Entità parametriche

Come nel caso delle entità interne, nelle estensioni del DTD può essere conveniente aggiungere la dichiarazione di entità parametriche da utilizzare per controllare l'inclusione o l'esclusione di sezioni marcate. Si osservi l'esempio, già mostrato in precedenza, in cui la macro '**`commentato`**' serve per controllare alcune sezioni marcate, stabilendone il tipo.

```
<!DOCTYPE linuxdoc SYSTEM
[
<!ENTITY pericolo "!!">
<!ENTITY posix    "POSIX">
<!ENTITY unix     "Unix">
<!ENTITY xwin     "X Window System">
<!ENTITY edizione "1999.12.31">
...
<!ENTITY % commentato "INCLUDE">
]>
...
...
1 + 2 = 3
<![%commentato;[
La matematica non è un'opinione.
]]>
...
...
```

### 6.10.4 Ultime annotazioni sulla dichiarazione del DTD

Si è visto in che modo inizia un sorgente SGML. Quello che non è ancora stato chiarito è che il tipo di documento deve essere stato dichiarato nel DTD, anche se ciò può sembrare ridondante. In effetti è necessario dire di cosa è composto il documento. Nel DTD potrebbe apparire un'istruzione come quella seguente:

```
<!ELEMENT linuxdoc O O ( article | report | book | letter ) >
```

In questo esempio, si comprende che non è necessario usare marcatori del tipo '**<linuxdoc>**' '**</linuxdoc>**' per delimitare il sorgente SGML. Infatti, la coppia di '**O**' afferma che queste sono opzionali. Invece, il tipo di documento '**linuxdoc**' deve contenere esattamente un elemento del tipo '**article**', oppure '**report**', oppure '**book**', o ancora '**letter**'.

Il sorgente SGML che fa riferimento al tipo di documento '**linuxdoc**' e che utilizza il formato definito dall'elemento '**article**', sarà composto schematicamente come segue:

```
<!DOCTYPE linuxdoc SYSTEM>
<article>
...
...
...
</article>
```

Un tipo di documento potrebbe essere definito in maniera diversa, per esempio nel modo seguente:

```
<!element miodoc - - ( sezione+ ) >
```

In questo caso, il documento può contenere solo elementi '**sezione**', ed è obbligatorio l'utilizzo dei marcatori per indicare l'inizio e la fine del tipo di documento.

```
<!doctype miodoc system>
<miodoc>
        <sezione>
        ...
        ...
        ...
</miodoc>
```

## 6.11 Mappe di sostituzione (shortref)

Fino a questo punto, si è vista la filosofia dell'SGML applicata alla struttura del documento e all'indipendenza rispetto alla piattaforma. L'analizzatore SGML standard, oltre che convalidare il documento in base al DTD, si occupa di rielaborare il sorgente SGML per generare un risultato intermedio, più facile da gestire per altri programmi di composizione.

In un certo qual modo, questo risultato intermedio può essere controllato, all'interno del DTD, attraverso la definizione di mappe di sostituzione, o *shortref*.

Con questo meccanismo, si punta normalmente ad attribuire significati speciali a simboli determinati, oltre che a controllare la spaziatura orizzontale e verticale del testo.

### 6.11.1 Dichiarazione e abbinamento delle mappe di sostituzione

La mappa di sostituzione definisce un abbinamento tra un simbolo e un'entità che ne prenderà il posto. L'esempio seguente è solo un pezzo ipotetico della dichiarazione di una mappa del genere.

```
<!SHORTREF miamappa
...
        "[" lsqb
        "]" rsqb
        "~" nbsp
        "_" lowbar
        "#" num
        "%" percnt
        "^" circ
        "{" lcub
        "}" rcub
        "|" verbar >
```

Dall'esempio si può osservare che alcuni simboli vengono sostituiti con le entità relative, indicate solo per nome, senza bisogno della e-commerciale e del punto e virgola finale. Questo fatto, di per sé, potrebbe sembrare assolutamente inutile dal punto di vista di SGML: se si può scrivere una parentesi quadra aperta, perché sostituirla automaticamente con la sua entità corrispondente. Il fatto è che il software utilizzato per la composizione, potrebbe attribuire un significato speciale a una parentesi quadra, mentre quello che si vuole nel testo SGML è che questa valga solo per quello che appare. In tal modo, chi scrive dovrebbe utilizzare necessariamente la macro '**`[`**' per non creare problemi al programma di composizione o di elaborazione successiva.

Nello stesso modo, attraverso la mappa di sostituzione, si può attribuire un significato completamente diverso alla parentesi quadra aperta: per assurdo, potrebbe diventare una parentesi graffa...

```
<!SHORTREF miamappa
...
        "[" lcub
        "]" rcub
...
        "{" lcub
        "}" rcub
        "|" verbar >
```

Volendo fare delle acrobazie, si può associare un simbolo a un'entità che poi si traduce in un marcatore. Si osservi l'esempio.

```
<!ENTITY formula1 '<formula>'>
<!ENTITY formula0 '</formula>'>
<!SHORTREF miamappa
...
        "[" formula1
        "]" formula0
...
        "{" lcub
        "}" rcub
        "|" verbar >
```

In questo modo, quando nel testo si utilizzano le parentesi quadre, ciò che si ottiene è l'apertura e la chiusura dell'elemento '**`formula`**'.

Anche se questa tecnica è stata usata nel noto DTD LinuxDoc, come in Qwertz, proprio per delimitare agevolmente le formule matematiche, si tratta di una cosa decisamente sconsigliabile dal punto di vista dell'SGML.

Gli elementi SGML vanno abbinati alle mappe che si ritiene siano più adatte per i loro scopi. Tuttavia, un elemento può non essere stato abbinato esplicitamente ad alcuna mappa; in tal caso eredita quella dell'elemento che lo contiene effettivamente, di volta in volta, nel documento. Di conseguenza, diventa importante abbinare esplicitamente una mappa almeno all'elemento più esterno, ovvero a quello che corrisponde al nome del tipo stesso di documento.

Dagli esempi mostrati, sarà stato notato che la mappa ha un nome, indicato subito dopo la parola chiave '**SHORTREF**' che apre il comando. Se si vuole abbinare la mappa '**miamappa**' all'elemento '**acronimo**', si procede come nell'esempio seguente:

```
<!USEMAP miamappa acronimo>
```

## 6.11.2 Spaziature e interruzioni di riga

In linea di principio, il risultato dell'elaborazione dell'analizzatore SGML contiene tutti gli spazi orizzontali e verticali esistenti nel sorgente di partenza. Però, per quanto possibile, si cerca normalmente di evitare che il sorgente SGML sia vincolato dalla spaziatura utilizzata, che in realtà potrebbe servire solo per facilitarne la lettura umana con rientri, allineamenti, spazi verticali come si farebbe con un linguaggio di programmazione.

Per questo ci deve essere un modo per poter identificare le spaziature orizzontali, le righe vuote e quelle bianche, in modo da poterle sopprimere nel risultato dell'elaborazione SGML. Naturalmente, bisogna poter distinguere, perché ci sono situazioni in cui gli spazi e le righe vuote hanno un significato e vanno mantenuti.

Per queste cose si utilizzano delle macro speciali, ma prima di descriverle, occorre definire alcuni concetti. Dal punto di vista dell'SGML, una riga è una sequenza di caratteri, con un inizio e una fine, ignorando completamente la codifica che si utilizza in pratica per separare una riga dall'altra.

Nei sistemi Unix, il codice di interruzione di riga è composto dal carattere *<LF>* mentre in altri sistemi si utilizza la sequenza *<CR><LF>*. Per l'SGML è come se questi codici non esistessero: le righe finiscono **prima** del codice di interruzione di riga e iniziano **dopo** tale codice. Si osservi l'esempio seguente:

```
<paragrafo>Ciao,
come stai?
Io bene; e tu?</paragrafo>
```

L'idea che ha l'SGML di ciò che è stato scritto, può essere rappresentata dallo schema seguente, dove è stato utilizzato il simbolo '**^**' per segnalare l'inizio della riga, il simbolo '**$**' per segnalarne la fine, i simboli '**>**' e '**<**' per indicare l'inizio e la fine dell'elemento.

```
                                >Ciao,$
                                ^come stai?$
                                ^Io bene; e tu?<
```

Può sembrare strano, ma all'inizio e alla fine del testo mancano questi margini: esiste solo l'inizio e la fine dell'elemento. Se si dovesse sopprimere una riga, si eliminerebbe implicitamente anche il suo inizio e la sua fine.

Da qualche parte si potrebbe leggere che il codice di inizio riga equivale al codice *<LF>*, mentre quello di fine riga corrisponde a *<CR>*. Evidentemente questo ragionamento può valere solo per

le piattaforme che utilizzano file di testo con un'interruzione di riga *<CR><LF>*, ma si tratta di una semplificazione che non corrisponde alla logica di SGML e può essere solo forviante.

La tabella 6.9 mostra le macro più importanti che possono essere usate per il controllo delle spaziature superflue.

Tabella 6.9. Simboli di definizione di spaziature e delimitazione delle righe.

| Simbolo | Significato |
|---|---|
| `&#RS;` | Inizio di una riga (*Record start*). |
| `&#RE;` | Fine di una riga (*Record end*). |
| `&#RS;B` | Spaziatura iniziale. |
| `B&#RE;` | Spaziatura finale. |
| `&#RS;&#RE;` | Una riga vuota. |
| `&#RS;B&#RE;` | Una riga contenente solo spazi orizzontali (bianca). |
| `&#SPACE;` | Uno spazio singolo, [ *SP* ]. |
| `&#TAB;` | Tabulazione, [ *HT* ]. |
| `BB` | Spazio orizzontale all'interno e agli estremi dell'elemento. |

L'esempio seguente mostra una mappa di sostituzione tipica, in cui si vogliono ignorare (e di conseguenza, eliminare) gli spazi orizzontali superflui, le righe vuote, quelle bianche, infine si vuole che tutto il testo si traduca in una riga sola.

```
<!shortref miamappa
        "BB"            space
        "&#RS;B"        null
        "B&#RE;"        space
        "&#RS;B&#RE;"   null
        "&#RS;&#RE;"    null
        "&#RS;"         null
        "&#RE;"         space
        "[" lsqb
        "]" rsqb
        "~" nbsp
        "_" lowbar
        "#" num
        "%" percnt
        "^" circ
        "{" lcub
        "}" rcub
        "|" verbar >
```

Le macro '**`&space;`**' e '**`&null;`**' si riferiscono rispettivamente a un solo carattere spazio e alla stringa nulla. Generalmente devono essere dichiarate nel DTD nel modo seguente:

```
<!ENTITY space " ">
<!ENTITY null "">
```

Per comprendere meglio l'effetto della mappa di sostituzione proposta, conviene partire da un esempio e analizzare gli effetti di ogni dichiarazione, una alla volta. In particolare, gli utenti dei sistemi Unix devono dimenticare per un po' il comportamento del codice di interruzione di riga (*newline*), perché SGML considera solo la stringa nulla all'inizio e alla fine della riga: solo quando la fine di una riga e l'inizio della successiva sono stati rimossi, allora queste due vengono unite assieme.

Supponiamo di cominciare da una variante dell'esempio già descritto, dove sono stati aggiunti tanti spazi orizzontali e verticali superflui.

```
>     Ciao,      $
^$
^         $
^come stai?$
^$
^     Io         bene;         e      tu?     <
```

Applicando la trasformazione '**"BB" space**', vengono sostituiti gli spazi orizzontali all'inizio dell'elemento, alla fine e all'interno delle frasi con uno spazio singolo normale.

```
> Ciao,      $
^$
^         $
^come stai?$
^$
^     Io bene; e tu? <
```

Si può osservare che le frasi si sono ricompattate; inoltre, all'inizio e alla fine dell'elemento è rimasto un solo spazio superfluo (che non potrà essere rimosso). Si continua applicando '**"&#RS;B" null**'; si ottiene l'eliminazione dell'inizio delle righe (quelle che contengono effettivamente qualcosa) fino al primo carattere diverso da uno spazio orizzontale.

```
> Ciao,      $
^$
^         $
^come stai?$
^$
 Io bene; e tu? <
```

Quando si applica anche '**"B&#RE;" space**'; si ottiene la sostituzione degli spazi orizzontali nella parte finale, fino alla fine delle righe (quelle che contengono effettivamente qualcosa), con uno spazio singolo. Nell'esempio, dal momento che nella prima riga è scomparso il simbolo che segnalava la fine della riga, appare un trattino basso, ma solo per aiutare il lettore.

```
> Ciao,_
^$
^         $
^come stai?$
^$
 Io bene; e tu? <
```

La sostituzione '**"&#RS;B&#RE;" null**' elimina le righe bianche, ma non vuote.

```
> Ciao,_
^$
^come stai?$
^$
 Io bene; e tu? <
```

La sostituzione '**"&#RS;&#RE;" null**' elimina le righe vuote.

```
> Ciao,_
^come stai?$
 Io bene; e tu? <
```

Si può osservare che la riga contenente la frase «come stai?», è rimasta intatta. Infatti, non contenendo spazi aggiuntivi all'inizio o alla fine, non è mai stata interessata dalle trasformazioni applicate fino a questo momento.

Finalmente entrano in gioco '**"&#RS; null**' e '**"&#RE; space**', per eliminare l'inizio e la fine delle righe rimaste. Per la precisione, la fine delle righe deve essere sostituito con uno spazio singolo, altrimenti si rischia di attaccare assieme delle parole. La trasformazione viene mostrata in due passaggi.

```
                                > Ciao,_
                                 come stai?_
                                 Io bene; e tu? <
```

```
                        > Ciao, come stai? Io bene; e tu? <
```

Nonostante la descrizione fatta con tanta cura, è probabile che la trasformazione '**"&#RS; null**' venga semplicemente ignorata, perché l'analizzatore SGML si limita a tenere in considerazione solo la fine delle righe (*record end*).

### 6.11.3 Limitazioni ed esagerazioni

Da quanto visto nella sezione precedente si potrebbe supporre che il meccanismo delle mappe di sostituzione permetta di sostituire quello che si vuole. Non è così, solo alcuni simboli sono considerati dei possibili *shortref*. In ogni caso, ci si accorge subito quando si usa qualcosa di sbagliato: l'analizzatore SGML avvisa immediatamente.

Attraverso le mappe di sostituzione si possono realizzare anche delle acrobazie che spesso sono poco giustificabili e che sarebbe meglio evitare. A parere di chi scrive, la cosa meno utile che si possa richiedere a un sistema SGML è quella di fare in modo che le righe vuote e quelle bianche nel sorgente siano trasformate in separazioni tra i paragrafi. Infatti, l'SGML non ha questo scopo, eppure molti sistemi si impegnano in questo senso. LinuxDoc raggiunge questo risultato intervenendo proprio nelle mappe di sostituzione, facendo in modo che le righe bianche, identificate dal simbolo '**&#RS;B&#RE;**', e quelle vuote, identificate dal simbolo '**&#RS;&#RE;**', siano sostituite da '** **', ovvero dai marcatori che servono a chiudere e a riaprire un paragrafo.

```
...
<!ENTITY psplit '</p><p>' >
...
<!SHORTREF pmap
        "&#RS;B" null
        "&#RS;B&#RE;" psplit
        "&#RS;&#RE;" psplit
...
        "{" lcub
        "}" rcub
        "|" verbar >
...
```

Quello che si vede sopra è proprio un estratto dal DTD di LinuxDoc, dove si vede che la macro '**&psplit;**' viene poi rimpiazzata dai marcatori già descritti.

Naturalmente, questo non esclude la possibilità di generare una grande quantità di elementi '**p**' vuoti, in presenza di più righe vuote o bianche. È chiaro che, successivamente, il sistema di composizione utilizzato deve prendersi carico della loro eliminazione.

### 6.11.4 Soluzione normale

Dopo aver visto in quanti modi si possono usare le mappe di sostituzione, vale la pena di mostrare una soluzione «normale», in cui il problema che si vuole risolvere è l'eliminazione degli spazi superflui all'inizio e alla fine delle righe, oltre che l'eliminazione delle righe bianche e quelle vuote:

```
<!ENTITY space " ">
<!ENTITY null "">
<!ENTITY recordstart "&#RS;">
<!ENTITY recordend "&#RE;">

<!SHORTREF standard
        "&#RS;B"        recordstart
        "B&#RE;"        recordend
        "&#RS;B&#RE;"   null
        "&#RS;&#RE;"    null
>
```

In questo modo, come si vede, è stato necessario dichiarare due entità nuove, '**recordstart**' e '**recordend**', per poter sopprimere gli spazi iniziali e finali superflui, pur mantenendo la separazione in righe distinte.

## 6.12 Elementi di testo riportato letteralmente

La predisposizione di un elemento SGML che consenta la scrittura di testo da riportare in modo letterale costituisce un problema. Si possono scegliere soluzioni diverse, ma nessuna perfetta secondo tutti i punti di vista.

Questo tipo di problema è particolarmente sentito nella scrittura di documenti tecnici, in cui ci può essere la necessità di mostrare porzioni di codice scritto in un qualche linguaggio di programmazione. Per evitare che simboli determinati vengano interpretati dall'analizzatore SGML, occorrerebbe utilizzare continuamente delle macro alternative.

Si possono seguire due direzioni per cercare di risolvere il problema: l'uso di elementi predisposti per un tipo di contenuto più o meno letterale, oppure l'uso di elementi normali con l'aggiunta di una sezione marcata di tipo '**CDATA**'.

### 6.12.1 Tipo di contenuto letterale

Nella definizione di un elemento occorre stabilite il tipo di contenuto. A livello elementare, quando l'elemento non può contenere altri elementi, si utilizza normalmente la parola chiave '**#PCDATA**', con cui si fa riferimento a testo che viene analizzato alla ricerca di entità generali da espandere, senza ammette altri elementi al suo interno.

Per ottenere un elemento adatto al contenuto letterale, si usa solitamente il tipo di contenuto definito dalla parola chiave '**RCDATA**', che non è perfettamente letterale, ma vi si avvicina molto. Per la precisione, la forma del testo viene mantenuta, con tutte le sue spaziature e le interruzioni di riga, ma le entità generali vengono espanse, mentre vengono ignorati eventuali marcatori di apertura. Ciò significa che, la e-commerciale ('**&**') non può essere usata in modo letterale, a meno di usare una macro adatta al suo posto. Lo stesso ragionamento riguarda la sequenza di minore e barra obliqua ('**</**'), che è ammessa solo nel marcatore di chiusura di questo elemento.

```
<!ELEMENT formattato - - RCDATA>
```

L'esempio mostra la dichiarazione dell'elemento '**formattato**', di tipo '**RCDATA**'. Generalmente, per poter utilizzare questo elemento nel modo corretto, si devono dichiarare anche due entità generali specifiche.

```
<!ENTITY ero   CDATA "&">
<!ENTITY etago '</' >
```

In tal modo, al posto del simbolo '**&**' si dovrà utilizzare la macro '**&ero;**', mentre al posto della sequenza '**</**', si dovrà usare la macro '**&etago;**'. È il caso di osservare che l'entità generale '**ero**' è volutamente diversa da un'entità analoga, necessaria a indicare una e-commerciale in un testo normale. Infatti, in questo caso, si vuole generare un testo letterale, che si presume possa essere interpretato nello stesso modo letterale anche da altro software di composizione successivo.

In alternativa, si potrebbe usare anche un tipo di contenuto definito dalla parola chiave '**CDATA**', che dovrebbe essere in grado di ignorare sia i simboli dei marcatori, che le macro delle entità generali. Di fatto però, questo tipo di elemento non dà normalmente i risultati sperati.

### 6.12.2 Sezioni marcate

Nel sorgente SGML, all'interno di un elemento che non sia stato predisposto per un contenuto letterale, è possibile inserire una sezione marcata di tipo '**CDATA**', come nell'esempio seguente:

```
<![CDATA[
Testo letterale: &amp;, &etago;, <ciao>, </ciao>,
ecc., vengono trattati in modo letterale.
]]>
```

In tal modo, vengono preservati anche gli spazi, orizzontali e verticali, e ogni eventuale mappa di sostituzione (*shortref*) viene ignorata temporaneamente. L'unica cosa che non può contenere questo ambiente, è la sequenza '**]]>**', che serve a concludere la sezione marcata.

Bisogna tenere presente che la sequenza '**]]>**' può essere rappresentata anche con l'inserzione di spazi; per esempio come '**] ]>**', '**]] >**' o '**] ] >**', che rappresentano sempre la stessa cosa.

Questa tecnica ha il vantaggio di potersi applicare anche a un DTD che non sia stato predisposto con elementi atti all'inserimento di testo letterale. Purtroppo, non tutti gli strumenti SGML sono in grado di riconoscere le sezioni marcate; si pensi ai navigatori, che pur sapendo interpretare l'HTML, non sono sempre in grado di riconoscere tali particolarità.

## 6.13 Cataloghi

Nelle sezioni precedenti si è visto che il DTD può essere composto da diversi file fisici nel sistema. Lo stesso preambolo di un sorgente SGML prevede la dichiarazione e l'inclusione di un DTD. È stato mostrato come includere un blocco di DTD esterno, attraverso la dichiarazione e il successivo utilizzo di un'entità parametrica che fa riferimento a un file esterno.

Quando si vogliono utilizzare componenti esterni senza fare riferimento a un file preciso, si possono predisporre dei cataloghi, con i quali si esplicitano questi dettagli riferiti al sistema di cui si dispone effettivamente.

Questo tipo di approccio viene usato tipicamente per due motivi: evitare di dover fare riferimento a un file preciso per il DTD nella dichiarazione del tipo di documento all'inizio del sorgente

SGML; includere in modo dinamico le entità standard riferite alle lettere accentate e ai simboli speciali. Per quanto riguarda il secondo problema, si deve tenere presente che l'SGML si astrae dalla piattaforma, quindi, il modo in cui le entità di questo tipo vanno rappresentate dipende da quello che si vuole fare dopo.

### 6.13.1 Riferimenti esterni

Generalmente, quando si vogliono usare i cataloghi, si possono fare due tipi di riferimenti a componenti esterne: l'identificatore pubblico e l'identificatore di sistema. Seguono quattro esempi significativi a questo proposito: nei primi due si tratta della dichiarazione del tipo di documento '**HTML**' e di un'entità parametrica, attraverso un identificatore pubblico (una stringa piuttosto lunga); negli ultimi due si tratta delle stesse dichiarazioni, ma fatte attraverso un identificatore di sistema.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

```
<!ENTITY % ISOlat1 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN">
```

```
<!DOCTYPE HTML SYSTEM>
```

```
<!ENTITY % ISOlat1 SYSTEM>
```

Si tratta, evidentemente, di due approcci equivalenti, ma che hanno delle conseguenze differenti nell'applicazione pratica. Dalle parole chiave utilizzate, '**PUBLIC**' e '**SYSTEM**', si può intuire che l'identificatore di sistema è legato alla situazione del sistema, anche se non è obbligatoria l'indicazione immediata del file corrispondente.

L'uso degli identificatori pubblici è quindi una scelta più conveniente, essendo meno vincolata alla piattaforma. Infatti, questi vengono utilizzati prevalentemente per tutto ciò che è già stato standardizzato: i DTD standard e le entità esterne standard.

### 6.13.2 Il catalogo in pratica

Quando si usano strumenti di analisi ed elaborazione SGML comuni, il catalogo è un file. A seconda degli strumenti utilizzati, potrebbe essere necessario configurare una variabile di ambiente, o usare un'opzione opportuna nella riga di comando, per comunicare a questi la sua posizione.

Il catalogo serve a esplicitare tutte le componenti esterne che non sono state indicate in modo preciso (il nome del file). Si osservi l'esempio seguente:

```
-- Entità standard richiamate attraverso un identificatore di sistema --
-- Sarebbe meglio non usare questo metodo --
ENTITY %ISOlat1            "ISOlat1"
ENTITY %ISOnum             "ISOnum"
ENTITY %ISOdia             "ISOdia"

-- Entità standard richiamate attraverso un identificatore pubblico --
-- Questo tipo di indicazione è preferibile in generale --
PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN"               "ISOlat1"
PUBLIC "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN"  "ISOnum"
PUBLIC "ISO 8879:1986//ENTITIES Diacritical Marks//EN"           "ISOdia"

-- DTD predefinito per il tipo HTML --
DOCTYPE "HTML"                                  "html32.dtd"

-- Identificatori pubblici per le varie forme dell'HTML 3.2 --
PUBLIC "-//W3C//DTD HTML 3.2//EN"               "html32.dtd"
PUBLIC "-//W3C//DTD HTML 3.2 Draft//EN"         "html32.dtd"
PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"         "html32.dtd"
```

Ogni direttiva dell'esempio occupa una riga e si compone di tre parti, dove l'ultima informazione rappresenta il file da utilizzare per quel particolare tipo di entità, documento o identificatore pubblico.

Per la precisione, invece che di file, occorrerebbe parlare di identificatore di sistema effettivo, dove questo concetto viene poi definito dallo standard ISO 8879. In generale si tratta di file e questo dovrebbe bastare come primo approccio all'SGML.

Si noti che i commenti sono delimitati da coppie di trattini, '`--`', come si fa all'interno delle istruzioni SGML.

| Direttiva | Descrizione |
|---|---|
| `PUBLIC` ***identificatore_pubblico identificatore_di_sistema*** | Stabilisce l'identificatore di sistema effettivo (il file) corrispondente all'identificatore pubblico indicato. Quando possibile, è preferibile utilizzare gli identificatori pubblici per definire gli oggetti. |
| `DOCTYPE` ***nome identificatore_di_sistema*** | Stabilisce l'identificatore di sistema effettivo (il file) corrispondente al nome del tipo di documento indicato. Dal momento che questo nome può fare riferimento a uno tra diversi DTD alternativi (si pensi al caso dell'HTML con le sue versioni), questa dichiarazione serve prevalentemente per stabilire un DTD predefinito nel caso in cui non sia stato specificato un identificatore pubblico nel documento che si elabora. |
| `ENTITY` ***nome identificatore_di_sistema*** | Stabilisce l'identificatore di sistema effettivo (il file) corrispondente all'entità generale indicata. |
| `ENTITY %`***nome identificatore_di_sistema*** | Stabilisce l'identificatore di sistema effettivo (il file) corrispondente all'entità parametrica indicata. Si osservi il fatto che il simbolo di percentuale è attaccato al nome dell'entità. |

Negli esempi seguenti, viene mostrata prima l'istruzione utilizzata nel DTD, o nel preambolo del sorgente SGML, quindi si presenta la direttiva corrispondente, necessaria nel catalogo.

- ```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
  ```
  Si tratta della dichiarazione, all'inizio di un sorgente SGML, dell'utilizzo del DTD '**HTML**', definito in base all'identificatore pubblico '**-//W3C//DTD HTML 3.2 Final//EN**'. Perché da questo si possa arrivare a identificare un file particolare, occorre che nel catalogo appaia una direttiva come quella seguente:

  ```
  PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"         "html32.dtd"
  ```
  In tal caso, all'identificatore pubblico '**-//W3C//DTD HTML 3.2 Final//EN**', viene abbinato il file '`html32.dtd`'.

- ```
  <!DOCTYPE HTML SYSTEM">
  ```
  Si tratta della dichiarazione, all'inizio di un sorgente SGML, dell'utilizzo del DTD '**HTML**', utilizzando un identificatore di sistema che non viene precisato in modo effettivo. Perché da questo si possa arrivare a identificare un file particolare, occorre che nel catalogo appaia una direttiva come quella seguente:

  ```
  DOCTYPE HTML            "html32.dtd"
  ```
  In tal caso, all'identificatore di sistema '**HTML**', viene abbinato il file '`html32.dtd`' (l'identificatore di sistema effettivo).

- ```
  <!ENTITY % ISOlat1 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN">
  ```
  All'interno del DTD, dichiara l'entità parametrica '**ISOlat1**' definita secondo l'identificatore pubblico '**ISO 8879:1986//ENTITIES Added Latin 1//EN**'. Perché da questo si possa arrivare a identificare un file particolare, occorre che nel catalogo appaia una direttiva simile a una delle due mostrate di seguito:

  ```
  PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN" "ISOlat1.latex"
  ```

  ```
  ENTITY %ISOlat1 "ISOlat1.latex"
  ```
  Nel primo caso, all'identificatore pubblico '**ISO 8879:1986//ENTITIES Added Latin 1//EN**', viene abbinato il file '`ISOlat1.latex`'; nel secondo, si specifica direttamente che l'entità parametrica '**ISOlat1**' corrisponde al contenuto del file '`ISOlat1.latex`'.

- ```
  <!ENTITY % ISOlat1 SYSTEM>
  ```
  Nel DTD, viene dichiarata l'entità parametrica '**ISOlat1**', utilizzando un identificatore di sistema che non viene precisato in modo effettivo. Perché da questo si possa arrivare a identificare un file particolare, occorre necessariamente che nel catalogo appaia una direttiva come quella seguente, dal momento che non è possibile fare riferimento a un identificatore pubblico:

  ```
  ENTITY %ISOlat1             "ISOlat1.latex"
  ```
  In questo modo si definisce l'identificatore di sistema effettivo dell'entità parametrica '**ISOlat1**', facendola corrispondere al file '`ISOlat1.latex`' (l'identificatore di sistema effettivo).

## 6.14 Riferimenti

- C. M. Sperberg-McQueen, Lou Burnard, *Guidelines for Electronic Text Encoding and Interchange (TEI P4)*, in particolare il secondo capitolo: *A gentle introduction to XML*

  *<http://www.tei-c.org/P4X/>*

- *The SGML Newsletter*

  *<http://architag.com/tag/>*

- *The SGML/XML Web Page*

  *<http://www.oasis-open.org/cover/>*

- *The SGML Centre*

  *<http://www.sgml.u-net.com/>*

# Analisi lessicale

Gli errori che si possono fare scrivendo un testo sono di vario tipo, ma quelli puramente lessicali, ovvero ciò che potrebbe essere classificato come errore di battitura, rappresentano i meno importanti. Tuttavia, si tratta pur sempre di una buona percentuale nell'insieme globale di errori che può contenere un testo.

Un programma banale che sia in grado di mostrare le parole che risultano semplicemente sconosciute, è già un buon aiuto verso l'obiettivo dello scrivere in modo corretto.

Un programma di analisi lessicale è utile quando si può gestire un dizionario personale, perché non si possono escludere le eccezioni da un testo, come per esempio il nome o il cognome di una persona, un indirizzo o una sigla particolare. In presenza di documenti di grandi dimensioni, diventa necessario gestire un dizionario specifico per ognuno di questi, in modo da non interferire con l'analisi di altri in cui certi termini, ammissibili da una parte, non possono esistere dall'altra.

## 7.1 Ispell

Ispell [1] è un programma di scansione lessicale che permette la realizzazione di dizionari contenenti anche indicazioni sulle possibili aggregazioni di parole (si pensi alla lingua tedesca in cui le parole sono generate spesso dall'unione di altre).

Lo studio di questa caratteristica di Ispell riguarda chi vuole realizzare un dizionario standard per un linguaggio particolare: generico o specifico di un certo settore. Qui si intende mostrare un uso semplificato di questo programma, in cui si utilizzano dizionari standard e si generano i propri dizionari personali specifici per ciò che si fa.

### 7.1.1 Dizionari

Generalmente, il pacchetto di distribuzione di Ispell contiene un dizionario standard per la lingua inglese. Dovrebbe trattarsi del file '`/usr/lib/ispell/english.hash`'. Nella stessa directory vanno collocati altri file per altre lingue, o per linguaggi specifici. Questi file, terminanti con l'estensione '`.hash`', sono ottenuti a partire da una coppia di file di testo, attraverso la compilazione con '**`buildhash`**', ogni volta che si cambia piattaforma.

È disponibile un pacchetto contenente un dizionario generico per la lingua italiana. Lo si dovrebbe trovare presso <*ftp://ftp.pluto.linux.it/pub/pluto/ildp/ispell/*> e si tratta di un file denominato secondo il modello '`italiano-`***`versione`***`.tgz`'.

Il dizionario italiano si compone di due file sorgenti: '`italiano.aff`' e '`italiano.sml`'. Il primo dei due contiene la tabella *affix*, che in pratica rappresenta una serie di regole sull'insieme dei caratteri ammissibili e sulla possibile unione di parti di parole, mentre il secondo è l'elenco di parole vero e proprio. Queste parole elencate, contengono a volte dei riferimenti aggiuntivi indicati dopo una barra obliqua ('`/`') che hanno valore in base alle definizioni della tabella *affix*. L'approfondimento sulla sintassi del file *affix* è utile solo se si vuole realizzare un dizionario hash specifico, mentre l'utilizzatore normale può ignorare questo problema. La compilazione dei file sorgenti in modo da ottenere un dizionario hash si ottiene con il comando seguente:

```
$ buildhash italiano.sml italiano.aff italiano.hash
```

Si otterrà il file '`italiano.hash`', da collocare nella directory '`/usr/lib/ispell/`'. Se si intende utilizzare sistematicamente questo dizionario, si può predisporre la variabile di ambiente

[1] **Ispell** software libero con licenza speciale

'**DICTIONARY**', assegnandovi il nome del file: '`italiano.hash`'. In alternativa, si può usare '**ispell**' con l'opzione '**-d**', come nell'esempio seguente (l'estensione '`.hash`' è predefinita e può essere omessa).

```
$ ispell -d italiano documento.txt
```

I dizionari personali sono invece una cosa diversa: si tratta di un elenco di termini, scritto con le stesse modalità di un sorgente, senza un file *affix* a fianco (o meglio, utilizzando quello del dizionario hash a cui si fa riferimento). Normalmente, tali file personali sono aggiornati da Ispell, quando questo viene usato in modo interattivo. Il nome predefinito del dizionario personale è '`~/.ispell_`***linguaggio***'. Per esempio, se si utilizza il dizionario standard predefinito, viene generato e utilizzato il file '`~/.ispell_english`' (nella directory personale), a meno di specificare un nome diverso con le opzioni.

In aggiunta ai file personali ci possono essere dei file più specifici, legati alla directory corrente: '`./.ispell_`***linguaggio***'. Inoltre, in mancanza dell'indicazione del linguaggio, i dizionari personali e quelli specifici hanno i nomi: '`~/.ispell_default`' e '`./.ispell_default`'.

## 7.1.2 Avvio e opzioni fondamentali

`ispell` [ ***opzioni*** ] ***file_da_analizzare***

Quella che si vede rappresenta una semplificazione estrema della sintassi dell'eseguibile '**ispell**', però, prima di apprendere il funzionamento delle particolarità di questo programma, è meglio comprendere le sue possibilità fondamentali.

Ispell può funzionare in modo interattivo, oppure no. In teoria, è possibile anche realizzare un programma che sfrutti le funzionalità di Ispell attraverso una pipeline; in pratica, si tratta in questo caso dell'utilizzo meno importante che si può fare di Ispell.

| Opzione | Descrizione |
|---|---|
| `-d` ***dizionario_hash*** | Permette di specificare un file dizionario differente da quello predefinito (che di solito è '`english.hash`'). Il nome del file viene indicato generalmente senza estensione e senza percorso, facendo implicitamente riferimento alla directory '`/usr/lib/ispell/`' e a file con estensione '`.hash`'. |
| `-p` ***dizionario_personale*** | Permette di specificare un dizionario personale differente da quello predefinito (che di solito è '`~/.ispell_`…'). |
| `-W` ***n_caratteri*** | Specifica la lunghezza delle parole che non devono essere prese in considerazione. In pratica, da quel numero di caratteri in giù, si considerano tutte valide. |
| `-x` | Evita la creazione di una copia di sicurezza. Senza indicare questa opzione, dovrebbe essere salvata una copia del file originale aggiungendo al suo nome l'estensione '`.bak`'. |
| `-b` | Si tratta dell'opzione opposta a '**-x**', in quanto permette di forzare la richiesta di creazione di una copia di sicurezza. |
| `-t` | Fa in modo che il testo da analizzare sia considerato un sorgente TeX, o LaTeX, per il quale si devono ignorare i codici di formattazione e possibilmente anche alcune indicazioni che sono solo funzionali a TeX, dal momento che non riguardano il contenuto del testo. Questa dovrebbe essere la modalità predefinita di funzionamento.<br>In generale, questa modalità va bene anche per il testo puro e semplice, purché non ci siano barre oblique inverse che possano essere confuse con comandi di TeX. |

| Opzione | Descrizione |
|---|---|
| `-n` | Fa in modo che il testo da analizzare sia considerato un sorgente Nroff o Troff, per il quale si devono ignorare i codici di formattazione.<br>La possibilità di distinguere i codici di formattazione di TeX, *roff, o altro, dipende anche dal file *affix* del dizionario utilizzato. |

### 7.1.3 Funzionamento interattivo

Il funzionamento normale di Ispell è interattivo. Generalmente viene fatta una copia di sicurezza del file analizzato, con un nome che termina con l'aggiunta dell'estensione '`.bak`', quindi Ispell permette di modificare il contenuto del file originale, in base alle scelte dell'utente.

Figura 7.1. Funzionamento interattivo di Ispell.

```
     stai               File: lettera


 Ciao come stai?

 00: stab     09: st-AI
 01: stag
 02: staid
 03: stain
 04: stair
 05: Stan
 06: star
 07: stay
 08: st AI

 [SP] <number> R)epl A)ccept I)nsert L)ookup U)ncap Q)uit e(X)it or ? for help
```

La figura 7.1 mostra il caso di un file, denominato '`lettera`', che contiene una frase normalissima, in cui la parola «stai» non viene riconosciuta. In effetti, si suppone di avere utilizzato il dizionario hash predefinito, ovvero quello inglese.

La parola '**`stai`**' viene evidenziata se le caratteristiche del terminale lo consentono; in ogni caso, viene indicata a parte, all'inizio (come si vede dall'esempio). Se possibile, Ispell elenca una serie di alternative possibili, in base alle affinità che può avere il termine sconosciuto con altre parole contenute nel dizionario. Questo elenco è numerato, in modo da permetterne la selezione. Nella parte bassa dello schermo appare un menù riepilogativo degli altri comandi a disposizione; comandi che si richiamano prevalentemente con la semplice pressione di tasti o combinazioni di tasti mnemonici.

| Comando | Descrizione |
|---|---|
| [ *Spazio* ] | Fa in modo che Ispell accetti la parola temporaneamente. Se ne troverà ancora, Ispell le segnalerà nuovamente. |
| [ *R* ] \| [ *r* ] | Richiede la sostituzione della parola errata con un'altra che deve essere inserita subito dopo. Se anche la nuova parola non sembra valida, questa viene segnalata ugualmente da Ispell. La sostituzione riguarda solo quell'occorrenza particolare; se verrà ritrovato ancora lo stesso errore, Ispell continuerà a segnalarlo. |
| [ *A* ] \| [ *a* ] | Fa sì che Ispell ignori la parola per tutto il resto del documento. |
| [ *I* ] \| [ *i* ] | Fa sì che Ispell accetti la parola e la inserisca nel dizionario personale, esattamente com'è, rispettando maiuscole e minuscole. |

| Comando | Descrizione |
|---|---|
| [ *U* ] \| [ *u* ] | Fa sì che Ispell accetti la parola e la inserisca nel dizionario personale, senza distinguere tra maiuscole e minuscole. |
| [ *0* ] \| [ *1* ] \| ... \| [ *0* ][ *0* ] \| [ *0* ][ *1* ] \| ... | La selezione di un numero fa riferimento alle voci proposte come parole alternative a quella errata. Con questa selezione di intende ottenere la sostituzione delle parole. È importante osservare che, se l'elenco supera le nove unità, la selezione avviene con due cifre numeriche. L'esempio che appare nella figura mostra questo caso: per indicare la parola '**stag**', occorre la sequenza [ *0* ][ *1* ]. |
| [ *X* ] \| [ *x* ] | Conclude il lavoro completando la scrittura del file e ignorando altri errori eventuali. Chiude anche il file del dizionario personale, mantenendo le voci aggiunte fino a quel punto. |
| [ *Q* ] \| [ *q* ] | Termina immediatamente, lasciando inalterato il file, senza conservare i termini eventualmente annotati per l'aggiunta nel dizionario personale. |
| [ *Ctrl+l* ] \| [ *Ctrl+L* ] | Ripulisce lo schermo. |

Per quanto riguarda il funzionamento interattivo di Ispell, sono importanti due opzioni.

| Opzione | Descrizione |
|---|---|
| `-M` | Richiede espressamente la visualizzazione del menù riassuntivo dei comandi interattivi. Di solito, tale menù appare in modo predefinito, a meno di avere compilato Ispell con opzioni particolari. |
| `-N` | Fa in modo che il menù riepilogativo dei comandi non venga visualizzato. |

Segue la descrizione di alcuni comandi.

- `$ ispell -d italiano lettera`

  Analizza il file '`lettera`' utilizzando il dizionario hash '**`italiano`**', ovvero, il file '`/usr/lib/ispell/italiano.hash`'.

- `$ ispell -d italiano -p mio lettera`

  Come nell'esempio precedente, ma in questo caso si utilizza il dizionario personale rappresentato dal file '`./mio`'. Nell'esempio precedente, si faceva riferimento al dizionario personale predefinito: '`~/.ispell_italiano`'.

### 7.1.4 Funzionamento non interattivo

Quando Ispell funziona in modo non interattivo, si limita a generare un elenco di termini, anche ripetuti, che risultano sconosciuti in base al dizionario. Ispell può anche essere utilizzato attraverso un altro programma, quando si indica l'opzione '**`-a`**', ma si tratta di un modo un po' complicato, che qui non viene descritto.

Per ottenere l'elenco dei termini sconosciuti, si utilizza l'opzione '**`-l`**'. Per esempio, questa possibilità di Ispell può essere sfruttata per produrre rapidamente un dizionario personale.

Se si dispone di un testo della cui esattezza si è certi, si può ottenere da Ispell l'elenco dei termini da lui sconosciuti, generando poi un dizionario personale con tutte queste eccezioni. Si procede nel modo seguente:

```
$ ispell -d italiano -l < romanzo > mio_dizionario
```

In questo modo, tutti i termini contenuti nel file '`./romanzo`' che non risultano dal dizionario hash '**`italiano`**', vengono emessi attraverso lo standard output e diretti nel file '`./mio_dizionario`'.

```
$ sort -f < mio_dizionario > dizionario1
```

In questo modo si riordina l'elenco di parole ottenuto, generando il file '`./dizionario1`', dove l'opzione '**`-f`**' serve a non distinguere tra lettere minuscole e maiuscole, anche se restano i doppioni. Con questo elenco si vuole generare un dizionario personale, eliminando questi doppioni ed eventualmente generando altre semplificazioni.

```
$ munchlist -s italiano -l italiano.aff dizionario1 > dizionario2
```

In questo modo, si ottiene il compattamento del file '`./dizionario1`', in base a quanto già contenuto del dizionario hash '**`italiano`**' e secondo le regole del file *affix* '`./italiano.aff`', generando il file '`./dizionario2`', che finalmente può essere utilizzato come dizionario personale.

In alternativa, si può anche tentare di dare in pasto a Ispell il file ottenuto dopo l'ordinamento, senza filtrarlo attraverso '**`munchlist`**'. Sarà Ispell stesso che eliminerà i doppioni.

### 7.1.5 Programmi di servizio di contorno a Ispell

Ispell si compone di diversi file binari. Il più importante è '**`ispell`**', come si è visto, ma altri sono necessari per la gestione dei file di dizionario. Si è già accennato a '**`buildhash`**' e a '**`munchlist`**', il cui utilizzo è il caso di riepilogare.

| |
|---|
| `buildhash` ***dizionario_sorgente file_affix dizionario_hash*** |
| `munchlist` [`-l` ***file_affix***] [`-s` ***dizionario_hash***] [***elenco_da_ridurre***] `>` ***elenco_ridotto*** |

Quelle mostrate sono le sintassi semplificate di questi due programmi. Di più può essere appreso dalla lettura di *ispell(1)*.

Segue la descrizione di alcuni esempi.

- `$ munchlist mio_dizionario > dizionario`

  Utilizza il dizionario hash e il file *affix* standard per ridurre l'elenco contenuto nel file '`./mio_dizionario`', generando il file '`./dizionario`'.

- `$ munchlist -s italiano -l ./italiano.aff mio_dizionario > dizionario`

  Utilizza il dizionario hash '**`italiano`**' ('`/usr/lib/ispell/italiano.hash`') e il file *affix* '`./italiano.aff`' per ridurre l'elenco contenuto nel file '`./mio_dizionario`', generando il file '`./dizionario`'.

- `$ buildhash italiano.sml italiano.aff italiano.hash`

  Genera il dizionario hash '`./italiano.hash`', a partire dall'elenco '`./italiano.sml`' e dal file *affix* '`./italian.aff`'.

### 7.1.6 Gestione dei dizionari personali

L'utilizzo occasionale di Ispell richiede la presenza di un dizionario hash e probabilmente di uno personale predefinito, che quasi sicuramente sarà '`~/.ispell_italiano`'. Ma la correzione ortografica basata esclusivamente su un dizionario è tanto più efficace quanto minore è il numero delle parole previste, ovvero, quanto più specifico è il dizionario utilizzato.

Di fronte alla realizzazione di un documento di un certo impegno, o di una serie di documenti che trattano dello stesso genere di cose, potrebbe essere conveniente utilizzare un dizionario personale specifico per quel progetto, eventualmente partendo da un dizionario hash praticamente vuoto.[2]

Per realizzare un dizionario «vuoto», adatto a qualunque linguaggio che utilizzi la codifica ISO 8859-1, si potrebbe partire dal file *affix* che contiene solo le righe seguenti, il cui unico scopo è quello di ammettere l'uso di tutte le lettere accentate e speciali.[3]

```
# minimo.aff
# Accetta qualunque carattere accentato e speciale di ISO 8859-1

wordchars       [a-z]   [A-Z]
wordchars       [à-\376]        [À-\336]
wordchars       [\337]
wordchars       [\377]

prefixes

suffixes
```

Le parole chiave '**`prefixes`**' e '**`suffixes`**' sono obbligatorie, ma il file è ancora incompleto (viene segnalato dai programmi come '**`buildhash`**' e '**`munchlist`**'), anche se funziona ugualmente per lo scopo che ci si prefigge qui.

Volendo esagerare, se le cifre numeriche possono avere un ruolo nella composizione delle parole che si vogliono controllare, si può aggiungere anche la riga seguente, tenendo conto che però poi '**`munchlist`**' non funziona tanto bene.[4]

```
wordchars       [0-9]
```

A fianco di questo si deve creare un elenco di parole che ne contenga almeno una, come nell'esempio seguente:

```
Linux
```

Si suppone che il file *affix* sia stato nominato '`minimo.aff`' e che l'elenco sia '`minimo.sml`'. Per creare il file hash, si procede come è già stato presentato più volte.

```
$ buildhash minimo.sml minimo.aff minimo.hash
```

Pur con una segnalazione di errore, dovuta all'estrema semplicità del file *affix*, si ottiene il file '`minimo.hash`' nella directory corrente. Questo file hash può essere usato solo per testi normali, senza codici di formattazione di alcun tipo, dal momento che il file *affix* mostrato non è stato predisposto per questo.

Se si dispone di un documento ritenuto sicuro, si può generare il dizionario personale relativo.

```
$ ispell -d ./minimo.hash -l < documento.txt > elenco
```

In questo modo si ottiene l'elenco delle parole usate nel file '`documento.txt`', che sono praticamente tutte sconosciute. Questo elenco deve essere riordinato e ridotto.

[2] Quando si ha a che fare con documentazione tecnica, in cui l'uso di termini in inglese è frequente, si potrebbe addirittura valutare la possibilità di basare l'analisi sul dizionario standard ('`english.hash`'), affiancando il dizionario personale specifico per il documento, solo che in tal caso si avrebbero difficoltà con le lettere accentate, dal momento che queste non sono previste nel file *affix* inglese.

[3] Le lettere '**ÿ**' e '**ß**', corrispondenti ai codici '**`\377`**' e '**`\337`**', sono minuscole e non hanno un equivalente maiuscolo nella codifica ISO 8859-1.

[4] In pratica, '**`munchlist`**' elimina queste parole ritenute estranee. Se si dispone di un elaboratore ben equipaggiato, si può dare in pasto a Ispell il file ottenuto dopo il riordino; sarà poi lui a eliminare i doppioni.

```
$ sort -f < elenco > elenco1
```

```
$ munchlist -l minimo.aff -s minimo.hash elenco1 > dizionario
```

Dopo la riduzione si ottiene finalmente il dizionario personale specifico del documento; successivamente si potranno eseguire le verifiche sullo stesso documento di origine (a seguito di aggiunte o di modifiche), con il comando seguente:

```
$ ispell -d ./minimo.hash -p ./dizionario documento.txt
```

# Analisi sintattica e stilistica con Textchk

L'analisi sintattica di un testo è un problema ben più complicato della semplice verifica delle parole con un dizionario. Esistono però alcuni tipi di errori sintattici, o stilistici, che si possono identificare con l'aiuto di espressioni regolari (*regular expression*).

La lingua italiana consente spesso l'utilizzo di forme espressive differenti, per le quali dovrebbe esserci almeno uniformità all'interno di uno stesso documento. Per esempio, occorre decidere se si vuole scrivere: «una aula» oppure «un'aula», «ed anche» oppure «e anche»,...

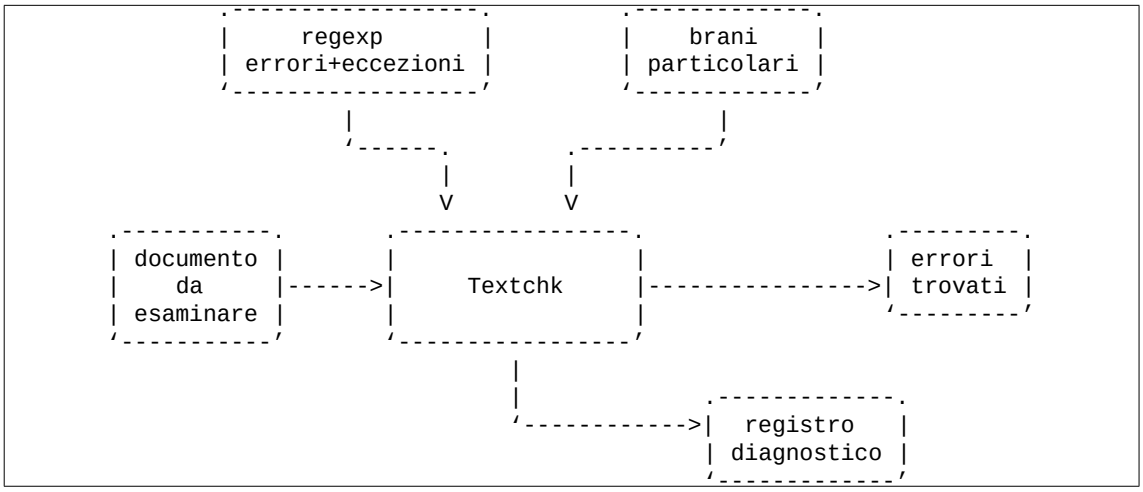
In questo capitolo si vuole mostrare un programma Perl che può aiutare a definire delle regole rappresentate in forma di espressioni regolari, per segnalare degli errori sintattici o stilistici. Con questo programma è possibile indicare anche delle regole di eccezione e delle particolarità riferite a un solo documento. Il programma in questione è Textchk, [1] che è derivato dagli strumenti preparati originariamente per la composizione di questo documento (ALtools e Alml).

Textchk dovrebbe trovarsi assieme alla distribuzione di questa opera; tuttavia, il suo riferimento principale è <*http://a2.swlibero.org/~daniele/software/textchk/*>.

## 8.1 Principio di funzionamento

Textchk scandisce un file di partenza generando un altro file contenente le parti di testo che risulterebbero errate (oltre a un file diagnostico contenente la registrazione del procedimento di verifica). Prima di iniziare a leggere il file da esaminare, vengono caricati dei modelli che esprimono degli errori, espressi in forma di espressione regolare, seguiti eventualmente da dei modelli di eccezione. Infine, vengono caricate anche delle particolarità riferite al testo che si elabora, trattate in forma letterale e non più secondo il modello di un'espressione regolare.

Figura 8.1. Schema di funzionamento di Textchk.




Gli errori che si possono ricercare attraverso delle espressioni regolari, riguardano la vicinanza di parole che hanno caratteristiche determinate, come l'uso o meno di articoli apostrofati. Sotto questo aspetto, diventa importante che, nel file di testo originale, ogni paragrafo si trovi su una sola riga, cioè non sia interrotto su più righe.

A fianco di questo problema, si aggiunge il fatto che il file sorgente che si vuole esaminare potrebbe contenere dei codici di controllo, come nel caso di TeX (o LaTeX) e di HTML. In tutte queste situazioni, prima di passare all'analisi vera e propria, occorre ripulire e riadattare il testo,

[1] **Textchk** GNU GPL

in modo da avere a che fare con un file di testo puro, in cui ogni paragrafo si trovi su una sola riga. Al limite, può essere sufficiente che ogni periodo, cioè ogni frase completa che termina con un punto, si trovi su una sola riga.

### 8.1.1 Espressioni regolari

Textchk è scritto in Perl, pertanto le espressioni regolari che possono essere gestite sono quelle di questo linguaggio di programmazione.

La ricerca della corrispondenza con le espressioni regolari che esprimono un errore, viene fatta in modo da circoscrivere, se possibile, tre parole prima e dopo della zona dell'errore. Per questa ragione, non ha senso tentare di identificare l'inizio e la fine di una riga (con i simboli '**^**' e '**$**'), inoltre non è possibile utilizzare le parentesi tonde.

A titolo di esempio, si propone il problema della «d» eufonica, per la precisione il caso di «ad». Supponendo di volerla utilizzare solo quando la parola successiva inizia con la vocale «a», escludendo il caso in cui la parola continui con un'altra «d» (per esempio: «ad amare», ma non «ad adattare»), si possono usare le espressioni regolari seguenti per individuare gli errori.

```
\ba\s+a[^d]\w*\b
\bad\s+ad\w*\b
\bad\s+[^a]\w*\b
```

Per intendere meglio il significato di ciò che è scritto, la prima riga significa:

| | |
|---|---|
| '**\b**' | lo spazio vuoto prima della parola; |
| '**a**' | la lettera «a»; |
| '**\s+**' | uno o più spazi orizzontali; |
| '**a[^d]**' | la lettera «a» seguita immediatamente da qualunque cosa che sia diversa dalla lettera «d»; |
| '**\w***' | zero o più caratteri alfabetici; |
| '**\b**' | lo spazio vuoto dopo la parola; |

Nello stesso tempo, però, si può decidere di accettare un'eccezione: «ad esempio», che secondo quando stabilito con l'ultima delle espressioni regolari appena mostrate, dovrebbe essere un errore. Si può usare quindi l'espressione regolare seguente, tra le eccezioni.

```
\bad\s+esempio\b
```

## 8.2 Configurazione

La configurazione di Textchk serve a definire gli errori sintattici che si ricercano. In generale è importante definire una configurazione specifica per ogni singolo progetto di documentazione, ma resta la possibilità di stabilire regole personali, legate all'utente, oltre che regole generali legate al sistema (per quanto questo possa avere un valore relativo).

La configurazione avviene attraverso un file di testo normale, in cui le righe bianche, quelle vuote e quelle che iniziano con il simbolo '#' vengono ignorate. Le altre righe sono dei record che possono avere una delle due forme seguenti:

```
DBL_____regola_di_errore [_____testo_esplicativo]
ERR_____regola_di_errore [_____testo_esplicativo]
EXC_____regola_di_eccezione
```

Nel primo caso si identifica una parola che si ritiene possa essere stata scritta due volte, in modo erroneo; il secondo indica un modello di errore, mentre nel terzo si tratta di un'eccezione. I record che descrivono le regole di eccezione si riferiscono sempre all'ultima regola di errore (di tipo '**DBL**' o '**ERR**') che sia stata incontrata fino a quel punto.

La forma di questi record è un po' strana, nel senso che la separazione dei campi avviene attraverso una sequenza di quattro trattini bassi ('____'). Ciò serve per evitare di creare problemi alla realizzazione delle espressioni regolari che descrivono gli errori e le eccezioni.

```
#--------------------------------------------------------------------
# d eufonica
# a|e|o prendono una «d» eufonica se sono seguite da una parola che
# inizia con la stessa vocale, a meno che ci sia subito dopo un'altra
# «d».
#--------------------------------------------------------------------
ERR____\ba\s+a[^d]\w*\b____a --> ad
EXC____\bda\s+a\s+a\b

ERR____\bad\s+ad\w*\b____ad --> a

ERR____\bad\s+[^aA]\w*\b____ad --> a
EXC____\bad\s+esempio\b
EXC____\bad\s+ora\b

ERR____\be\s+e[^d]\w*\b____e --> ed
ERR____\bed\s+[eE]d\w*\b____ed --> e
ERR____\bed\s+[^eèE]\w*\b____ed --> e

ERR____\bo\s+[oO][^d]\w*\b____o --> od
ERR____\bod\s+[oO]d\w*\b____od --> o
ERR____\bod\s+[^oO]\w*\b____od --> o
```

L'esempio mostra una serie di istruzioni con le quali si cerca di definire l'uso della «d» eufonica. Vale la pena di analizzare cosa succede di fronte a una situazione precisa. Si suppone di avere scritto un testo nel quale è stata inserita la frase seguente:

```
Purtroppo, fino ad ora il colore dell'auto non è stato scelto dal cliente.
```

Concentrando l'attenzione sui record di configurazione seguenti, si può simulare ciò che succede.

```
ERR____\bad\s+[^aA]\w*\b____ad --> a
EXC____\bad\s+esempio\b
EXC____\bad\s+ora\b
```

Per cominciare, viene individuato un errore in via preliminare in corrispondenza di «ad ora», perché la parola che segue «ad» non inizia con una lettera «a». Textchk preleva una stringa di tre parole prima e tre parole dopo questo errore: «Purtroppo, fino ad ora il colore dell'auto». In questo caso, le parole precedenti sono solo due, perché non è stato possibile ottenere di più.

Su questa stringa estratta viene condotto il controllo per le eccezioni successive; così, dal momento che si ottiene una corrispondenza (sempre con «ad ora»), l'errore si rivela infondato (in base ai presupposti stabiliti).

L'ultimo campo dei record che descrivono gli errori serve per indicare una spiegazione per ciò che viene identificato come un errore. Questa spiegazione viene mostrata da Textchk nel momento in cui l'errore relativo viene mostrato, secondo lo schema seguente:

***testo_esplicativo***
***tre_parole_precedenti>>errore<<tre_parole_successive***

Come si vede, la corrispondenza con l'errore viene evidenziata dai delimitatori '**>>**' e '**<<**'.

### 8.2.1 Gerarchia della configurazione

Textchk è stato pensato originariamente per avere una configurazione specifica per ogni progetto di documentazione che ogni autore possa gestire. Tuttavia, è possibile definire anche una configurazione personale e una di sistema. Si tratta dei file seguenti:

| | |
|---|---|
| '`./.textchk.rules`' | contiene la configurazione corrente, che viene letta prima delle altre; |
| '`~/.textchk.rules`' | contiene la configurazione personale, letta subito dopo quella corrente; |
| '`/etc/textchk.rules`' | contiene la configurazione di sistema, che viene letta alla fine. |

In generale non è opportuno stabilire una configurazione generale di sistema. Tuttavia, se c'è la necessità di annullare l'effetto di una regola di errore stabilita a livello generale, si può dichiarare la stessa regola nella configurazione personale o in quella corrente, facendola seguire immediatamente da un'eccezione identica. In pratica, supponendo di avere definito a livello di sistema la regola seguente, che richiede l'uso della «d» eufonica ogni volta che la parola seguente inizia con una vocale,

```
ERR____\b[aeo]\s+[aeiouAEIOU]\w*\b____a/e/o --> ad/ed/od
```

per annullarne l'effetto completamente, basta aggiungere la stessa regola in qualità di eccezione, subito dopo:

```
# Regola di sistema che qui viene annullata.
ERR____\b[aeo]\s+[aeiouAEIOU]\w*\b____a/e/o --> ad/ed/od
EXC____\b[aeo]\s+[aeiouAEIOU]\w*\b
```

### 8.2.2 Casi particolari

Alle volte non conviene indicare troppe eccezioni, oppure non è materialmente possibile. Per esempio, si può immaginare il caso in cui si vuole mostrare veramente un modo sbagliato di scrivere per qualche ragione. Per queste situazioni viene in aiuto un file di configurazione aggiuntivo, che però può essere associato esclusivamente a un solo progetto di documentazione. Si tratta del file '`./.textchk.special`', in cui si possono inserire integralmente alcune stringhe che Textchk ha indicato precedentemente come errate.

Per questa parte della configurazione non c'è molto da fare: basta utilizzare un programma per la creazione e la modifica dei file di testo ricopiando ciò che serve dal file che viene generato da Textchk per registrare gli errori trovati. L'esempio seguente mostra un estratto di quello che potrebbe contenere questo file. Si osservi il fatto che si tratta di esempi di errori scritti così di proposito.

```
oppure «un'aula», «ed anche» oppure «e
vuole scrivere: «una aula» oppure «un'aula»,
ma non «ad adattare»), si possono
```

### 8.2.3 L'indicazione di parole doppie

Un errore frequente nella scrittura di un testo consiste nella ripetizione di una parola per due volte di seguito, mentre l'intenzione era quella di scriverla una volta sola. Per intercettare questo tipo di situazione si utilizza il record '**DBL**'. Nel campo dell'espressione che indica l'errore, si fa riferimento implicitamente a una parola intera. Infatti, nella comparazione reale, viene aggiunto il simbolo '**\b**' all'inizio e alla fine, a sottolineare che la parola deve essere completa. Si osservi l'esempio seguente:

```
#-----------------------------------------------------------------------
# Parole doppie.
#-----------------------------------------------------------------------
DBL____\w\w+____Due parole identiche
EXC____\bciao\s+ciao\b
```

L'intenzione è di individuare qualunque parola ('**\w+**'), composta almeno da due caratteri, che si ripete immediatamente. Viene posta una sola eccezione alla coppia «ciao ciao».

## 8.3 Come si usa

Textchk si compone di un eseguibile unico, '**textchk**', che si utilizza secondo lo schema sintattico seguente:

| `textchk --input-type=`***tipo_di_file*** ***file_da_analizzare*** [***errori_risultanti*** [***file_diagnostico***]] |
|---|
| `textchk --help` |
| `textchk --version` |

Oltre alle opzioni standard, '**--help**' e '**--version**', l'opzione '**--input-type**' serve a stabilire il tipo di file che si fornisce in ingresso, in modo che Textchk sappia come fare per gestirlo opportunamente, attraverso un argomento:

| | |
|---|---|
| `--input-type=standard` | si riferisce a un file di testo in cui ogni capoverso occupa esattamente una riga e non richiede altri adattamenti; |
| `--input-type=man` | si riferisce a un file Troff delle pagine di manuale, che come tale richiede una rielaborazione in modo da ottenere un file di testo, simulando uno schermo di ampiezza orizzontale smisurata; |
| `--input-type=texinfo`<br>`--input-type=texi` | si riferisce a un sorgente Texinfo; |
| `--input-type=html` | si riferisce a un file HTML che può essere trasformato in un file di testo attraverso Lynx. |

Il secondo argomento della riga di comando è il nome del file da analizzare, che deve corrispondere al tipo indicato precedentemente. Il terzo argomento serve a definire il nome del file che viene creato per annotare le stringhe errate che vengono individuate; se non viene fornito espressamente il suo nome, viene creato un file con lo stesso nome di quello in ingresso, con l'aggiunta dell'estensione '`.err`' ('***file_da_analizzare***`.err`'). Il quarto argomento serve a specificare il nome del file diagnostico, nel quale vengono registrate tutte le fasi di individuazione di errori e di eccezioni. Anche l'indicazione di questo file può essere omessa; in tal caso viene usato il nome del file degli errori con l'aggiunta dell'estensione '`.diag`', oppure il file in ingresso con la stessa aggiunta ('***errori_risultanti***`.diag`' oppure '***file_da_analizzare***`.diag`').

Per esempio, il comando seguente genera i file '`bash.1.err`' e '`bash.1.diag`':

```
$ textchk --input-type=man bash.1
```

### 8.3.1 Come vengono mostrati gli errori e i dati diagnostici

Durante il suo lavoro, Textchk mostra sullo schermo ciò che trova, delimitando gli errori tra i delimitatori '**>>**' e '**<<**'. Per esempio, in base alle regole seguenti,

```
ERR____\bad\s+[^aA]\w*\b____ad --> a
EXC____\bad\s+esempio\b
EXC____\bad\s+ora\b
```

si possono ottenere segnalazioni come queste:

```
ad --> a
    Pertanto, andando >>ad elevare<< il proprio livello
ad --> a
    contrario, riuscendo così >>ad esplorare<< il proprio mondo
```

Nel file che elenca gli errori si trovano le righe seguenti:

```
Pertanto, andando ad elevare il proprio livello
contrario, riuscendo così ad esplorare il proprio mondo
```

Inoltre, nel file diagnostico si trova l'intero procedimento:

```
??? Pertanto, andando >>ad elevare<< il proprio livello
ERR \bad\s+[^aA]\w*\b
!!! Pertanto, andando >>ad elevare<< il proprio livello

??? contrario, riuscendo così >>ad esplorare<< il proprio mondo
ERR \bad\s+[^aA]\w*\b
!!! contrario, riuscendo così >>ad esplorare<< il proprio mondo

??? Il colore rosso, >>ad esempio<<, rappresenta la propria
ERR \bad\s+[^aA]\w*\b
EXC \bad\s+esempio\b

??? Il colore rosso, >>ad esempio<<, rappresenta la propria
ERR \bad\s+[^aA]\w*\b
EXC \bad\s+esempio\b

??? Pertanto, l'espressione «>>ad emettere<<» non è corretta.
ERR \bad\s+[^aA]\w*\b
SPC Pertanto, l'espressione «ad emettere» non è corretta.
```

Il file diagnostico mostra informazioni diverse, distinte attraverso una sigla iniziale. Le righe che iniziano con '**???**' indicano il problema trovato; le righe che iniziano con '**ERR**' rappresentano la regola di errore in base alla quale viene evidenziato il problema; le righe che iniziano con '**EXC**' indicano una regola di eccezione per la quale il problema viene superato; le righe che iniziano con '**SPC**' rappresentano un caso particolare (speciale), per cui la frase in questione viene accettata così come si trova. Infine, le righe che iniziano con '**!!!**' rappresentano la conferma finale che si deve trattare di un errore.

## 8.4 Come si installa

Textchk si compone di un solo programma Perl: '**`textchk`**'. Questo file può essere collocato ovunque sia ritenuto più conveniente, preferendo evidentemente una directory elencata all'interno della variabile di ambiente '**`PATH`**'.

Trattandosi di un programma Perl, deve essere disponibile l'interprete relativo. Attualmente si prevede che questo corrisponda esattamente all'eseguibile '`/usr/bin/perl`'. Se il proprio sistema non è organizzato in questo modo, basta modificare la prima parte del programma:

```
#!/usr/bin/perl
#...
```

Dopo la soluzione di questo problema, c'è solo bisogno di predisporre un file di regole, '`./.textchk.rules`', poi, mano a mano che il lavoro procede, potrà essere conveniente predisporre anche il file '`./.textchk.special`'.

### 8.4.1 Gettext

I messaggi che può mostrare Textchk possono essere tradotti, dal momento che viene usato il modulo Perl-gettext. Nel pacchetto del sorgente è presente un file di messaggi per la lingua italiana, che però deve essere compilato e installato:

```
$ msgfmt -o textchk.mo it.po
```

In questo modo, si genera il file '`textchk.mo`', che probabilmente va collocato nella directory '`/usr/share/locale/it/LC_MESSAGES/`'.

### 8.4.2 Dipendenze

Per funzionare, Textchk richiede l'interprete Perl e la presenza di un modulo speciale: Perl-gettext. Inoltre, per poter gestire correttamente i diversi tipi di file per cui è stato predisposto, richiede in particolare Groff, Lynx e Texinfo.

## 8.5 Riferimenti

- Daniele Giacomini, *Textchk*

  <*http://a2.swlibero.org/~daniele/software/textchk/*>

# Alml: preparazione e visione generale

Alml [1] è il sistema di composizione SGML di questo documento, *Appunti di informatica libera*. Si tratta di un programma Perl, '**`alml`**', che controlla l'analizzatore SGML e altri programmi necessari per arrivare alla composizione finale del documento. Tuttavia, per poter comprendere quanto esposto, è necessario prima conoscere ciò che è stato descritto a proposito dell'SGML, di TeX e dei sistemi comuni di composizione basati sull'SGML.

> Alml, con il suo DTD, continuerà a evolversi assieme all'opera *Appunti di informatica libera*. Chi desidera utilizzare questo sistema di composizione deve tenere in considerazione tale dinamicità; pertanto, prima di passare a un eventuale aggiornamento, deve valutare l'opportunità del cambiamento.

Alml si avvale di altri programmi per l'analisi SGML e per la generazione di alcuni formati finali. In particolare, è necessario disporre di '**`nsgmls`**' che fa parte generalmente del pacchetto SP (anche se la propria distribuzione GNU/Linux potrebbe nominarlo in modo differente); inoltre è fondamentale la presenza di LaTeX per generare la composizione da stampa. La tabella 9.1 riepiloga gli applicativi da cui dipende il buon funzionamento di Alml.

Tabella 9.1. Applicativi da cui dipende Alml.

| Applicativo | Compito |
|---|---|
| Perl | Alml è scritto in Perl. |
| Perl-gettext | Modulo Perl per l'utilizzo di Gettext. |
| SP | Verifica la validità SGML e genera una prima conversione. |
| LaTeX | Compone in un formato finale per la stampa. |
| PSUtils | Riorganizza, ingrandisce e riduce un file PostScript. |
| Dvipdfm | Consente una conversione in PDF a partire dal file DVI. |
| Uuencode | Estrae le immagini incorporate da file esterni. |
| ImageMagick | Converte i file delle immagini nei formati appropriati, adattando le dimensioni. |
| Ghostscript | Serve a ImageMagick per la conversione di file PostScript in altri formati. |
| HTML2ps | Consente l'importazione di codice HTML con LaTeX e con pdfLaTeX. |
| Links | Converte un file HTML in testo puro. |
| Lilypond | Consente l'importazione di codice Lilypond. |
| Xfig | Consente l'importazione di codice Xfig. |

## 9.1 Installazione di Alml

Alml viene fornito attraverso archivi tradizionali di tipo tar+gzip, in file con nomi del tipo:

```
alml-versione.tar.gz
```

Estraendo il contenuto dell'archivio, si dovrebbero ottenere in particolare i file e le sottodirectory elencati nella tabella 9.2, che rappresentano l'essenziale.

[1] **Alml** GNU GPL

Tabella 9.2. Contenuto essenziale dell'archivio di distribuzione di Alml.

| File o directory | Descrizione |
|---|---|
| 'bin/*' | File eseguibili. |
| 'doc/*' | Esempi e documentazione eventuale. |
| 'etc/' | File di configurazione da inserire a partire dalla directory '/etc/'. |
| 'man/*' | Pagine di manuale relative agli eseguibili. |
| 'share/sgml/*' | File e directory da collocare in '/usr/share/sgml/alml/'. |

Gli eseguibili, che nel pacchetto di distribuzione si trovano nella directory 'bin/', devono essere raggiungibili attraverso il percorso di ricerca del sistema, rappresentato dalla variabile di ambiente '**PATH**'. Pertanto vanno collocati opportunamente, oppure vanno predisposti dei collegamenti adeguati.

Quanto contenuto nella directory 'share/sgml/', va collocato nella directory '/usr/share/sgml/alml/', oppure vanno realizzati dei collegamenti equivalenti.

In generale, se la propria distribuzione GNU/Linux non è predisposta per la gestione delle entità standard ISO 8879, conviene modificare il collegamento simbolico 'alml.cat', che nella sua collocazione finale deve trovarsi nella directory '/usr/share/sgml/alml/'. Normalmente questo punta al file 'alml.cat.debian', ma in caso di problemi conviene modificarlo in modo che punti a 'alml.cat.normal'.

### 9.1.1 Gettext

I messaggi di Alml possono essere tradotti. Se si dispone del file PO relativo alla lingua preferita, è necessario compilarlo come nell'esempio seguente:

```
$ msgfmt -vvvv -o alml.mo it.po
```

In questo esempio, il file 'it.po' viene compilato generando il file 'alml.mo'. Trattandosi evidentemente della traduzione italiana, questo file può essere collocato in '/usr/share/locale/it/LC_MESSAGES/', o in un'altra posizione analoga in base agli standard del proprio sistema operativo.

Se non è disponibile il modulo Perl-gettext,[2] che serve a Alml per accedere alle traduzioni, è possibile eliminare il suo utilizzo e simulare la funzione di Gettext. In pratica si commentano le istruzioni seguenti all'inizio dei programmi '**alml**', '**alml-sp2sp**', '**alml-sp2be**' e '**alml-extra**':

```
# We *don't* want to use gettext.
#use POSIX;
#use Locale::gettext;
#setlocale (LC_MESSAGES, "");
#textdomain ("alml");
```

Inoltre, si tolgono i commenti dalla dichiarazione della funzione fittizia '**gettext()**', come si vede qui:

```
sub gettext
{
    return $_[0];
}
```

[2] Nelle distribuzioni Debian si tratta del pacchetto '**liblocale-gettext-perl**'.

## 9.2 Esempio iniziale

Un esempio iniziale può servire per comprendere il funzionamento generale di Alml.

```
<!DOCTYPE ALML PUBLIC "-//Daniele Giacomini//DTD Alml//EN">

<alml lang="it" spacing="uniform">
<head>
    <admin>
        <description>Un esempio per l'utilizzo del sistema Alml</description>
        <keywords>SGML, XML, HTML, Alml</keywords>
    </admin>
    <title>Esempio di utilizzo di Alml</title>
    <author>Pinco Pallino &lt;pinco.pallino@brot.dg&gt;</author>
    <date>2011.11.11</date>
    <legal>
        <p>Copyright &copy; Pinco Pallino, &lt;pinco.pallino@brot.dg&gt;</p>

        <p>Permission is granted to copy, distribute and/or modify this
        document under the terms of the GNU Free Documentation License,
        Version 1.1 or any later version published by the Free Software
        Foundation; with no Invariant Sections, with no Front-Cover
        Texts, and with no Back-Cover Texts. A copy of the license is
        included in the section entitled "GNU Free Documentation
        License".</p>
    </legal>
    <maincontents levels="2">Indice generale</maincontents>
</head>
<intro>
<h1>
Introduzione al documento
</h1>

<p>Questo documento è scritto per... bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla...</p>

<p>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla...</p>

</intro>
<body>
<h1 id="capitolo-primo">
Lavorando con bla bla bla...
<indexentry>lavorare con bla bla</indexentry>
<indexentry>bla bla</indexentry>
</h1>

<p>Lavorare con bla bla è molto semplice... bla bla bla bla bla bla bla
bla bla bla bla bla bla...</p>

<p>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla...</p>

<h2>
Fare di meglio
</h2>

<p>C'è anche un modo migliore per... bla bla bla bla bla bla bla bla bla
bla bla bla bla...</p>

<p>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla...</p>

<h1 id="capitolo-secondo">
Non dover lavorare più
```

```
<indexentry>relaxing</indexentry>
</h1>

<p>Se non si lavora ci si può riposare, ma questo si può fare solo se si
dispone già di una certa disponibilità economica... bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla...</p>

<p>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla...</p>

</body>
<appendix>
<h1>
Alcune note
</h1>

<p>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla...</p>

<p>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla...</p>

</appendix>
<index>
<h1>
Index
</h1>

<printindex index="main">

</index>
</alml>
```

Se tutto viene copiato correttamente nel file ipotetico ‘`esempio.sgml`’, con il comando seguente si ottiene la composizione in PostScript, attraverso LaTeX e Dvips:

```
$ alml --ps esempio.sgml
```

Con il comando seguente, si ottiene la composizione in HTML, su più file distinti:

```
$ alml --html esempio.sgml
```

## 9.3 Cosa si genera con la composizione

L’utilizzo di Alml può generare file differenti a seconda del tipo di operazione che viene richiesta. La tabella 9.3 riepiloga questi file.

Tabella 9.3. File generati dall’utilizzo di Alml. I file ‘`nome.sgml`’ e ‘`nome.css`’ devono essere già presenti.

| File | Descrizione |
|---|---|
| ‘*nome*`.sgml`’ | Il sorgente SGML principale da cui hanno origine gli altri file. |
| ‘*nome*`.css`’ | Foglio di stile CSS necessario per la composizione HTML. |
| ‘*nome*`.X2V.ps`’ | Composizione in PostScript con l’opzione ‘**`--long`**’. |
| ‘*nome*`.X3V.ps`’ | Composizione in PostScript con l’opzione ‘**`--extralong`**’. |
| ‘*nome*`.X3H.ps`’ | Composizione in PostScript con l’opzione ‘**`--large`**’. |
| ‘*nome*`.X4H.ps`’ | Composizione in PostScript con l’opzione ‘**`--extralarge`**’. |
| ‘*nome*`.X1T.ps`’ | Composizione in PostScript con l’opzione ‘**`--thin`**’. |

| File | Descrizione |
|---|---|
| '***nome*.aux' | File ausiliario e temporaneo della composizione attraverso LaTeX. |
| '***nome*.diag' | File diagnostico generato da '**alml**'. |
| '***nome*.pageref' | File temporaneo con i riferimenti alle pagine nella composizione con LaTeX. |
| '***nome*.dvi' | Composizione in DVI, finale o transitoria. |
| '***nome*.log' | File diagnostico generato da LaTeX. |
| '***nome*.pdf' | Composizione in PDF. |
| '***nome*.ps' | Composizione in PostScript. |
| '***nome*.tex' | Composizione transitoria in formato LaTeX. |
| '***nome*.html' | Primo file della composizione in HTML. |
| '***nomen*.html' | ***n***-esimo file della composizione in HTML. |
| '***n*.jpg' | ***n***-esimo file delle immagini relativo alla composizione in HTML. |
| '***n*.ps' | ***n***-esimo file delle immagini relativo alla composizione in PostScript o PDF. |
| '*~' | File temporaneo non meglio precisato. |

È bene sottolineare che i file indicati come '***nome*.sgml**' e '***nome*.css**' devono essere già presenti perché si possa usare Alml; inoltre, il sorgente SGML principale potrebbe a sua volta incorporare altri file SGML.

Se il sorgente SGML fa riferimento a immagini collocate in file esterni, è necessario che queste siano in formato PNG.[3] In generale, conviene prevedere una directory apposita per questi file, in modo da non essere intralciati quando la composizione in HTML, o in PostScript, genera la copia delle immagini richieste nella directory corrente, utilizzano i nomi nella forma '***n*.jpg**' o '***n*.ps**'.

Alle volte si possono incontrare problemi inspiegabili nell'inserimento di immagini, che si possono manifestare in modo particolare nella composizione in PDF. Spesso si superano questi problemi usando ImageMagick e facendo un passaggio intermedio nel formato JPG. Per esempio, disponendo del file 'pippo.png' che risulta corretto e perfettamente visibile con gli strumenti normali, ma che si comporta in modo strano nella composizione PDF, può convenire il passaggio seguente:

```
$ convert pippo.png pippo.jpg
$ convert pippo.jpg pippo.png
```

Al termine, il file 'pippo.jpg' può essere eliminato.

## 9.4 Sintassi nell'uso del programma frontale

Il programma frontale attraverso cui si gestisce il sistema di composizione Alml è '**alml**':

```
alml opzioni sorgente_sgml
alml --help
alml --version
```

[3] A seconda del tipo di composizione finale, può darsi che sia necessario convertire le immagini in un altro formato. In questi casi, viene usato ImageMagick per generare automaticamente ciò che serve. Per la precisione, il formato PNG di partenza è ciò che serve per la composizione in PDF; per la composizione in PostScript servono immagini EPS; per la composizione HTML vengono generati file in formato JPG.

Come si vede dal modello sintattico, a parte i casi delle opzioni '**--help**' e '**--version**', è sempre richiesta l'indicazione di un file sorgente SGML, a cui applicare un qualche tipo di elaborazione.

| Opzione | Descrizione |
|---|---|
| `--help` | Mostra la guida rapida interna e conclude il funzionamento. |
| `--version` | Mostra le informazioni sulla versione e conclude il funzionamento. |
| `--paper={a4\|letter}` | Permette di specificare le dimensioni della carta in base a un nome standard. Il formato predefinito è A4, che corrispondente alla parola chiave '**a4**'. |
| `--paper-orientation={normal\|inverted}` | Permette di specificare l'orientamento della carta. Si osservi che non vengono usate le definizioni tipiche, corrispondenti a *portrait* e *landscape*, perché qui il contesto è un po' diverso. A ogni modo, se si seleziona il formato di carta A4 e poi si aggiunge l'opzione '**--paper-orientation=inverted**', si intende arrivare a una composizione in orizzontale. |
| `--draft` | Quando il contesto lo permette, serve per ottenere una composizione particolare, con più informazioni utili alla correzione o alla revisione del testo. A differenza di quanto si potrebbe essere portati a pensare, in questo modo l'elaborazione è più complessa del normale, proprio per portare in risalto tali informazioni. |
| `--compact` | Quando il contesto lo permette, serve per ottenere una composizione compatta, risparmiando spazio. |
| `--long` | Quando si usa in abbinamento all'opzione '**--ps**', cioè quando si vuole ottenere un risultato in formato PostScript, permette di ottenere una composizione speciale, a due colonne con il testo ridotto della metà. Di solito si abbina a questa anche l'opzione '**--compact**'.<br>Si osservi che le opzioni '**--long**', '**--extralong**', '**--large**', '**--extralarge**' e '**--thin**', sono delle estensioni al formato selezionato con '**--paper**', che possono essere disponibili solo con pochi formati di partenza. Oltre a questo, va considerato il fatto che si possono utilizzare solo quando l'orientamento richiesto è di tipo «normale», tenendo conto che il risultato finale potrebbe essere orizzontale o verticale, in base al tipo di estensione. |

| Opzione | Descrizione |
|---|---|
| `--extralong` | Come '**`--long`**', su tre colonne verticali, molto rimpicciolite. |
| `--large` | Come '**`--long`**', su tre colonne in orizzontale. |
| `--extralarge` | Come '**`--large`**', su quattro colonne in orizzontale, molto rimpicciolite. |
| `--thin` | Un formato ridotto che si traduce in pratica in un A7x4, ovvero mezzo A4 diviso in verticale. |
| `--clean` | Rimuove alcuni file temporanei abbinati al file sorgente indicato. Si tratta per la precisione di '***nome***`.pageref`', '***nome***`.diag`', '***nome***`.aux`' e '***nome***`.log`'. |
| `--verbose` | Segnala il procedere dell'elaborazione con informazioni dettagliate. In generale tali informazioni sono ottenibili dal file '***nome***`.diag`'; tuttavia, in presenza di file sorgenti di grandi dimensioni, può servire per sapere a che punto è l'elaborazione. |
| `--sgml-include=`***entità_parametrica*** | Attraverso questa opzione, che può essere usata anche più volte, è possibile «includere» delle entità parametriche. Per la precisione, è come se nel sorgente venisse dichiarata un'entità parametrica corrispondente, assegnandole la parola chiave '**`INCLUDE`**'. Ciò viene usato per controllare l'inclusione di porzioni di sorgente, secondo le convenzioni dell'SGML. |
| `--page-numbering=`{`plain`\|`default`\|`tome`} | Questa opzione permette di definire in che modo gestire la numerazione delle pagine nei formati di composizione cartacei. In condizioni normali, la numerazione è realizzata attraverso sequenze differenti: una per la parte iniziale fino alla fine dell'introduzione, una per il corpo (comprese le appendici) e una finale per gli indici analitici. Assegnando la parola chiave '**`plain`**' si fa in modo che la numerazione sia unica, cosa che potrebbe essere conveniente per il formato PDF. Nel caso particolare della parola chiave '**`tome`**', si ottiene una numerazione separata dei volumi, con la conseguenza che alcuni indici, a seconda del contesto, oltre a indicare la pagina aggiungono un prefisso corrispondente al numero del volume in cui si trova. |
| `--sgml-syntax` \| `--sgml-check` | Una qualunque di queste due opzioni permette di ottenere la verifica formale del sorgente, in base al DTD. |

| Opzione | Descrizione |
|---|---|
| `--sp` | Con questa opzione si vuole raggiungere solo un formato intermedio per il controllo diagnostico del funzionamento di Alml. |
| `--tex` \| `--latex` | Con questa opzione si vuole raggiungere solo un formato intermedio in LaTeX per il controllo diagnostico del funzionamento di Alml. |
| `--dvi` | Genera un risultato in formato DVI. L'elaborazione crea una serie di file EPS per le immagini, secondo il modello '***n***`.ps`'. |
| `--ps` \| `--postscript` | Genera un risultato in formato PostScript. L'elaborazione crea una serie di file EPS per le immagini, secondo il modello '***n***`.ps`'; una volta ottenuto il file PostScript finale, questi file non servono più. |
| `--pdf` | Genera un risultato in formato PDF. |
| `--html` | Genera un risultato in formato HTML, articolato in più file, dove il primo è '***nome***`.html`' e gli altri sono '***nome n***`.html`'. Inoltre, viene fatta una copia dei file delle immagini, secondo il modello '***n***`.jpg`' (le due numerazioni sono indipendenti). |
| `--html-text` | Genera un risultato in formato HTML speciale, in un file unico, senza riferimenti a immagini esterne e con tabelle testuali. Il file ottenuto può essere consultato con Links e con questo può essere convertito in un testo puro e semplice, attraverso il comando: '**`links -dump` *nome*`.html > ` *nome*`.txt`**' |
| `--html-check` \| `--html401-check` | Se sono stati installati i file necessari, consente la verifica formale di un file HTML secondo le specifiche della versione 4.01. |
| `--html320-check` | Se sono stati installati i file necessari, consente la verifica formale di un file HTML secondo le specifiche della versione 3.2. |
| `--xml-check` | Se sono stati installati i file necessari, consente la verifica formale di un file XML secondo le specifiche del DTD relativo (attualmente solo XHTML). |

## 9.5 Organizzare un file-make

Un file-make opportuno può facilitare l'uso di Alml. Viene proposto un esempio elementare, riferito al file '`example.sgml`', in cui si può vedere anche l'utilizzo proposto di '**`alml`**'.

```
# file name prefix.
DOC_PREFIX=example

# Notice that "text" generates an HTML file with the same name
# for the first HTML page. This is why it is before the standard
# HTML typesetting.
#
all:    \
clean   \
text    \
html    \
ps      \
longps  \
extralongps  \
largeps  \
extralargeps  \
pdf

clean:
        @echo "Cleaning..."                              ; \
        find . -name core    -exec rm -f \{\} \;         ; \
        rm -f $(DOC_PREFIX)*.tex                         ; \
        rm -f $(DOC_PREFIX)*.dvi                         ; \
        rm -f $(DOC_PREFIX)*.sp                          ; \
        rm -f $(DOC_PREFIX)*.sp2                         ; \
        rm -f $(DOC_PREFIX)*.ps                          ; \
        rm -f $(DOC_PREFIX)*.pdf                         ; \
        rm -f $(DOC_PREFIX)*.txt                         ; \
        rm -f $(DOC_PREFIX)*.log                         ; \
        rm -f $(DOC_PREFIX)*.aux                         ; \
        rm -f $(DOC_PREFIX)*.tmp                         ; \
        rm -f $(DOC_PREFIX)*.diag                        ; \
        rm -f $(DOC_PREFIX)*.pageref                     ; \
        rm -f $(DOC_PREFIX)*.pageloc                     ; \
        rm -f *.html                                     ; \
        rm -f *.bak                                      ; \
        rm -f *.jpg                                      ; \
        rm -f *.ps                                       ; \
        rm -f *\~

check:
        @alml --sgml-check              \
              --verbose                 \
              $(DOC_PREFIX).sgml

dvi:
        @alml --dvi                     \
              --verbose                 \
              $(DOC_PREFIX).sgml

ps:
        @alml --ps                      \
              --verbose                 \
              $(DOC_PREFIX).sgml

longps:
        @alml --ps                      \
              --verbose                 \
              --compact                 \
```

```
              --long                    \
              --page-numbering=plain    \
              $(DOC_PREFIX).sgml

extralongps:
        @alml --ps                      \
              --verbose                 \
              --compact                 \
              --extralong               \
              --page-numbering=plain    \
              $(DOC_PREFIX).sgml

largeps:
        @alml --ps                      \
              --verbose                 \
              --compact                 \
              --large                   \
              --page-numbering=plain    \
              $(DOC_PREFIX).sgml

extralargeps:
        @alml --ps                      \
              --verbose                 \
              --compact                 \
              --extralarge              \
              --page-numbering=plain    \
              $(DOC_PREFIX).sgml

pdf:
        @alml --pdf                     \
              --verbose                 \
              --page-numbering=plain    \
              $(DOC_PREFIX).sgml

html:
        @alml --html                    \
              --verbose                 \
              $(DOC_PREFIX).sgml

text:
        @alml --html-text               \
              --verbose                 \
              $(DOC_PREFIX).sgml        ; \
        links -dump                      \
             $(DOC_PREFIX).html         \
              > $(DOC_PREFIX).txt
```

Si può osservare in particolare l'obiettivo '**`clean`**' che elimina tutti i file non indispensabili e in particolare tutti i file il cui nome termina per '`.html`' e per '`.ps`'.

Se per esempio si utilizza il comando '**`make ps`**', si otterrà la composizione in PostScript, generando in particolare il file '`example.ps`'.

## 9.6 Particolarità del sistema Alml

Recentemente, Alml è stato modificato e alcune sue funzionalità particolari, non sono più disponibili.

**Non esiste più la gestione delle derivazioni**, pertanto, non si possono più usare strutture come quelle seguenti:

```
<!-- START derivazione -->
…
…
<!-- STOP derivazione -->
```

Gli elementi '**verbatimpre**', '**asciiart**' e '**uri**', vanno usati assieme a una sezione marcata di tipo CDATA:

```
<verbatimpre>
<![CDATA[
…
…
…
]]>
</verbatimpre>
```

```
<asciiart>
<![CDATA[
…
…
…
]]>
</asciiart>
```

```
<uri><![CDATA[indirizzo]]></uri>
```

## 9.7 Usare Textchk, Checkbot e Ispell con Alml

Textchk e Checkbot, descritti rispettivamente nel capitolo 8 e nella sezione 5.4, possono essere usati facilmente con Alml. In generale, si passa per una composizione in formato HTML singolo, quindi si utilizzano questi programmi. Supponendo di avere generato il file '`mio_file.html`':

```
$ textchk --input-type=html mio_file.html mio_file.tchk mio_file.tdiag
```

```
$ checkbot --url file://'pwd'/mio_file.html
```

Per usare Ispell, è conveniente generare prima una versione del documento in formato testo puro. Per questo si potrebbe usare Lynx o Links, ma all'interno del pacchetto di Alml è disponibile un programma di supporto speciale, in grado di convertire opportunamente un file HTML per questo scopo. Si tratta di '**alml-extra**' che va usato con l'opzione '**--html-to-text-for-spell**':

```
alml-extra --html-to-text-for-spell < file_html > file_testo_non_formattato
```

In particolare, per evitare problemi con Ispell, nel file che si ottiene sono eliminate le barre oblique inverse ('\').

Naturalmente, usando poi Ispell nel file generato in questo modo, non ha senso fare delle correzioni, che invece vanno applicate al sorgente originale, in modo manuale.

## 9.8 Espandere le potenzialità elaborative di TeX

Il file LaTeX generato da Alml tende a richiedere risorse impreviste a TeX. È molto probabile che per documenti di dimensioni medie, sia necessario espandere i limiti posti dalla configurazione di TeX.

In generale, si dovrebbe disporre di una distribuzione teTeX, per la quale si interviene nel file '`texmf/web2c/texmf.cnf`' (eventualmente potrebbe trattarsi meglio di '`/etc/texmf/texmf.cnf`', o simile).

Per la composizione di *Appunti di informatica libera* si è resa necessaria la modifica di alcune variabili; quello che si vede sotto sono i valori minimi da assegnare alle variabili rispettive:[4]

```
main_memory = 2000000
font_mem_size = 800000
pool_size = 250000
hash_extra = 10000
buf_size = 100000
save_size = 40000
```

Si può tenere in considerazione l'abbinamento seguente, tra il rapporto generato da TeX e il file di configurazione '`texmf.cnf`', tenendo conto che in situazioni particolari il programma può segnalare la mancanza di una risorsa differente da quelle comuni:

```
Here is how much of TeX's memory you used:
```

- ```
  42853 strings out of 55918
  ```
  Dipende dalla variabile '**`max_strings`**'. In questo caso gli era stato assegnato il valore 60000.

- ```
  510063 string characters out of 647843
  ```
  Dipende dalla variabile '**`pool_size`**'. In questo caso gli era stato assegnato il valore 700000.

- ```
  200381 words of memory out of 1000001
  ```
  Dipende dalla variabile '**`main_memory`**'. In questo caso gli era stato assegnato il valore 1000000.

- ```
  44744 multiletter control sequences out of 10000+40000
  ```
  Il valore finale che si somma a 10000, dipende dalla variabile '**`hash_extra`**', a cui era stato assegnato il valore 40000.

- ```
  221835 words of font info for 188 fonts, out of 400000 for 1000
  ```
  I due valori finali dipendono rispettivamente da '**`font_mem_size`**' e da '**`font_max`**'.

- ```
  14 hyphenation exceptions out of 1000
  ```
  Dipende dalla variabile '**`hiph_size`**' a cui corrisponde esattamente il valore finale.

Al termine delle modifiche a questo file, occorre ricordare di lanciare il comando '**`texconfig init`**', con i privilegi dell'utente '**`root`**':[5]

```
# texconfig init
```

[4] La distribuzione GNU/Linux Debian organizza la configurazione del file '`texmf.cnf`' attraverso un insieme di file più piccoli, come verrà descritto più avanti.

[5] Non tutte le modifiche che si apportano a questo file richiedono l'esecuzione di '**`texconfig init`**'; tuttavia è meglio ripeterlo, anche per quelle situazioni in cui non serve.

Nel caso particolare della distribuzione Debian, il file di configurazione '`/etc/texmf/texmf.cnf`' è ottenuto attraverso la fusione di file differenti, contenuti nella directory '`/etc/texmf/texmf.d/`'. In tal caso, per modificare le voci descritte in precedenza, occorre intervenire probabilmente nel file '`/etc/texmf/texmf.d/95NonPath`'; successivamente occorre eseguire il comando '**`update-texmf`**', il quale ricostruisce un file '`/etc/texmf/texmf.cnf`' nuovo; infine si deve eseguire '**`texconfig init`**'.

### 9.8.1 Limiti strutturali di TeX

Le distribuzioni normali di TeX potrebbero non essere in grado di gestire un gran numero di comandi '**`\label`**', anche se si tenta di intervenire nella configurazione. Questo si traduce in pratica in un limite insuperabile per ciò che nella configurazione viene mostrato come la variabile '**`save_size`**'.

I comandi '**`\label`**' generano delle annotazioni in un file con estensione '`.aux`', simili all'esempio seguente:

```
\newlabel{anchor7}{{}{25}}
```

In questo caso si afferma che l'etichetta '**`anchor7`**' corrisponde alla pagina 25.

Generalmente, la composizione con i programmi '**`*tex`**' viene ripetuta per tre volte, allo scopo di acquisire le informazioni contenute in questo file: la prima volta viene costruito da zero, la seconda volta il testo viene reimpaginato utilizzando queste informazioni, rigenerandole nuovamente; infine, la terza volta non ci dovrebbero essere ulteriori spostamenti nell'impaginazione e il procedimento termina. Pertanto, la seconda e la terza volta viene letto il file con estensione '`.aux`'.

Sia i comandi '**`\label`**', sia i comandi '**`\newlabel`**' contenuti nel file ausiliario che viene incluso automaticamente, vanno a ridurre la memoria definita dalla variabile '**`save_size`**'. Così succede normalmente che si riesce a completare la prima elaborazione del file, mentre nella successiva, caricando anche il file ausiliario la memoria non basta più. La segnalazione di errore tipica è la seguente:

```
! TeX capacity exceeded, sorry [save size=40000].
```

Di fatto, questa variabile non può superare il valore 65535, anche se si tenta di modificare i sorgenti di teTeX intervenendo nel file '`texk/web2c/tex.ch`'. Dovrebbe esserci una riga simile a quella seguente:

```
@!inf_save_size = 600;
@!sup_save_size = 40000;
```

Si può anche provare, aumentando il valore assegnato a '**`sup_save_size`**', per esempio come nel caso seguente, ma in pratica, il limite massimo che si riesce a raggiungere resta quello di 65535:[6]

```
@!inf_save_size = 600;
@!sup_save_size = 100000;
```

[6] Il limite strutturale sembra dipendere da un'organizzazione del programma pensata per l'elaborazione su architetture a 16 bit.

### 9.8.2 Soluzione attuata da Alml

Alml è un sistema di composizione pensato per la realizzazione di opere molto grandi, con indici generali e analitici gestiti autonomamente. In questo modo, la composizione tradizionale attraverso TeX genererebbe un file '`.aux`' con una quantità di voci molto grande. Per evitare di saturare il limite di TeX, questi riferimenti vengono inseriti in un altro file, con estensione '`.pageref`' e gestiti esternamente a TeX.

In breve, Alml gestisce le cose nel modo seguente.

1. Viene creato un file TeX in cui le etichette (le ancore) usano il comando '**`\AlmlLabel`**':

   ```
   \AlmlLabel{etichetta}
   ```

   Inoltre, i riferimenti alle pagine si fanno con comandi del tipo:

   ```
   \AlmlPageRef{0}{000}{etichetta}
   ```

2. Viene avviato TeX che elabora il file e genera un file '`.pageref`' in base ai comandi '**`\AlmlLabel`**'.

3. Viene letto il file '`.pageref`' e con quelle informazioni, il file TeX viene modificato intervenendo sui riferimenti alle pagine, che diventano:

   ```
   \AlmlPageRef{1}{pagina}{etichetta}
   ```

4. Si riavvia TeX che genera un nuovo file '`.pageref`'.

5. Viene letto il file '`.pageref`' e, con quelle informazioni, il file TeX viene modificato intervenendo sui riferimenti alle pagine, che diventano:

   ```
   \AlmlPageRef{2}{pagina}{etichetta}
   ```

6. Si riavvia TeX per l'ultima volta.

### 9.8.3 Suddivisione automatica in tomi e parti della composizione finale PostScript e PDF

Per facilitare la suddivisione della composizione PostScript in file contenenti solo un tomo o solo una parte, vengono inserite nel sorgente TeX delle istruzioni per creare un file con estensione '`.pageloc`', contenente le informazioni necessarie:

```
BOF
tome{1}pageoffset{12}relativepage{1}
part{1}pageoffset{12}relativepage{7}
part{2}pageoffset{12}relativepage{19}
part{3}pageoffset{12}relativepage{105}
part{4}pageoffset{12}relativepage{121}
part{5}pageoffset{12}relativepage{171}
part{6}pageoffset{12}relativepage{203}
part{7}pageoffset{12}relativepage{269}
part{8}pageoffset{12}relativepage{319}
part{9}pageoffset{12}relativepage{351}
part{10}pageoffset{12}relativepage{383}
part{11}pageoffset{12}relativepage{411}
part{12}pageoffset{12}relativepage{415}
part{13}pageoffset{12}relativepage{469}
tome{2}pageoffset{12}relativepage{541}
part{14}pageoffset{12}relativepage{545}
eof{}pageoffset{12}relativepage{552}
EOF
```

Il significato dovrebbe essere intuitivo. Per esempio, il primo tomo inizia dalla 13-esima pagina (ottenuta sommando 12 a 1) e termina all'inizio del tomo successivo, ovvero alla 552-esima pagina (541+12-1). L'ultima pagina è la 564-esima.

In questo esempio, il valore 12 ricorrente rappresenta le pagine che precedono il contenuto vero e proprio del documento, in cui ci possono essere indici generali e introduzioni. Questo valore, definito qui come *page offset*, viene semplicemente sommato a quello finale.

### 9.8.4 Programma di supporto

Alml dispone di un programma di supporto, costituito dall'eseguibile '**`alml-extra`**', che consente di facilitare lo svolgimento di funzioni accessorie, in particolare per la riorganizzazione dei file PostScript.

`alml-extra` ***opzione*** [***argomento***]

A seconda dell'opzione utilizzata, può essere richiesto un argomento o meno, che fa riferimento a un file.

Dal momento che le opzioni che riguardano la conversione di file PostScript sono piuttosto difficili da ricordare, è disponibile anche uno script molto semplice che ne facilita l'uso:

`alml-extra-menu` ***file_ps***

Le opzioni che vengono descritte nel seguito si riferiscono a '**`alml-extra`**', usato direttamente.

| Opzione | Descrizione |
|---|---|
| `--help` | Mostra la guida rapida interna e conclude il funzionamento. |
| `--version` | Mostra le informazioni sulla versione e conclude il funzionamento. |

| Opzione | Descrizione |
|---|---|
| `--ps-group-pages=`***n_pagine*** | Prevede che l'argomento finale sia un file PostScript, in cui vengono modificate le stringhe di definizione delle pagine, in modo che si possano individuare raggruppamenti di ***n*** pagine, di solito per facilitare la rilegatura. In pratica, in questo modo, si individuano più facilmente le pagine che compongono una segnatura. |
| `--ps-renumber-pages` | Prevede che l'argomento finale sia un file PostScript, in cui vengono modificate le stringhe di definizione delle pagine, in modo che la sequenza sia rinumerata a partire da uno. |
| `--alml-ps-split-tome=`***file_posizione_pagine*** | Prevede che l'argomento finale sia un file PostScript, generato attraverso Alml, per il quale sia disponibile un file contenente la posizione di inizio dei vari tomi (dovrebbe trattarsi di un file con estensione '`.pageloc`'), che va indicato come argomento dell'opzione stessa. Quello che si ottiene sono diversi file PostScript, con estensione '`.`***n***`.ps`', dove in particolare '`.0.ps`' contiene le pagine precedenti al primo tomo effettivo, con la presenza eventuale di file con estensione '`.app.ps`' e '`ndx.ps`', per le pagine delle appendici e degli indici analitici rispettivamente. |
| `--alml-dvi-split-tome=`***file_posizione_pagine*** | Funziona come '**`--alml-ps-split-tome`**', ma si riferisce a file DVI. |

| Opzione | Descrizione |
|---|---|
| `--alml-ps-split-part=`***file_posizione_pagine*** | Prevede che l'argomento finale sia un file PostScript, generato attraverso Alml, per il quale sia disponibile un file contenente la posizione di inizio dei vari tomi (dovrebbe trattarsi di un file con estensione '`.pageloc`'), che va indicato come argomento dell'opzione stessa. Quello che si ottiene sono diversi file PostScript, con estensione '`.`***n***`.ps`', dove in particolare '`.0.ps`' contiene le pagine precedenti alla prima parte effettiva, con la presenza eventuale di file con estensione '`.app.ps`' e '`ndx.ps`', per le pagine delle appendici e degli indici analitici rispettivamente.<br>Se il documento che si va a suddividere prevede una suddivisione in tomi, la scomposizione che si ottiene non è perfetta, perché la fine di una parte che precede un tomo, si trova a includere le pagine che rappresentano l'inizio del tomo stesso, fino alla pagina che precede la parte successiva. |
| `--alml-dvi-split-part=`***file_posizione_pagine*** | Funziona come '**`--alml-ps-split-part`**', ma si riferisce a file DVI. |
| `--html-index=`***directory*** | Genera, attraverso lo standard output, un file HTML che potrebbe essere utilizzato come file '`index.html`', contenente un elenco molto semplice dei file contenuti nella directory indicata. |
| `--html-index-basic=`***directory*** | Come '**`--html-index`**', senza mostrare le date dei file. |
| `--html-index-basic-recursive` | Genera una serie di file '`index.html`', a partire dalla directory corrente e in tutte le sottodirectory. |
| `--html-to-text-for-spell` | Legge lo standard input, che dovrebbe essere costituito da un file HTML, filtrandolo allo scopo di generare un file di testo puro, utilizzabile per un controllo ortografico di qualche tipo. Il file che si ottiene viene emesso attraverso lo standard output. |
| `--perl-to-gettext` | Legge lo standard input, che dovrebbe essere costituito da un file sorgente Perl, filtrandolo allo scopo di generare un file di testo, adatto all'analisi da parte di Gettext, che solitamente riconosce bene solo le stringhe del linguaggio C. Il file che si ottiene viene emesso attraverso lo standard output. |

| Opzione | Descrizione |
|---|---|
| `--a4-to-a5-2-a4` | Prevede che l'argomento finale sia un file PostScript, in formato A4, che viene rielaborato in modo da ottenere due pagine A5 per ogni pagina A4 finale. Si ottiene un file con estensione '`.a5-2-a4.ps`'. |
| `--a4-to-a6-4-a4` | Prevede che l'argomento finale sia un file PostScript, in formato A4, che viene rielaborato in modo da ottenere quattro pagine A6 per ogni pagina A4 finale. Si ottiene un file con estensione '`.a6-4-a4.ps`'. |
| `--a4-to-a5-2-a4-1h-1` | Prevede che l'argomento finale sia un file PostScript, in formato A4, che viene rielaborato in modo da ottenere due pagine A5 per ogni pagina A4 finale, che deve essere piegata a metà, in orizzontale, rilegando il tutto a segnature di un solo foglio. Si ottiene un file con estensione '`.a5-2-a4-1h-1.ps`'. |
| `--a4-to-a5-2-a4-1h-10` | Prevede che l'argomento finale sia un file PostScript, in formato A4, che viene rielaborato in modo da ottenere due pagine A5 per ogni pagina A4 finale, che deve essere piegata a metà, in orizzontale, rilegando il tutto a segnature di 10 fogli. Si ottiene un file con estensione '`.a5-2-a4-1h-10.ps`'. |
| `--a4-to-a6-4-a4-2h-2` | Prevede che l'argomento finale sia un file PostScript, in formato A4, che viene rielaborato in modo da ottenere quattro pagine A6 per ogni pagina A4 finale, che deve essere piegata a metà, in orizzontale, per due volte, rilegando il tutto a segnature di due fogli. In pratica, ogni segnatura si ottiene da un solo foglio A4 che viene piegato due volte. Si ottiene un file con estensione '`.a6-4-a4-2h-2.ps`'. |
| `--a4-to-a6-4-a4-2h-4` | Prevede che l'argomento finale sia un file PostScript, in formato A4, che viene rielaborato in modo da ottenere quattro pagine A6 per ogni pagina A4 finale, che deve essere piegata a metà, in orizzontale, per due volte, rilegando il tutto a segnature di quattro fogli. In pratica, ogni segnatura si ottiene da due fogli A4 che vengono piegati assieme per due volte. Si ottiene un file con estensione '`.a6-4-a4-2h-4.ps`'. |

| Opzione | Descrizione |
|---|---|
| `--a4-to-a6-4-a4-2h-6` | Prevede che l'argomento finale sia un file PostScript, in formato A4, che viene rielaborato in modo da ottenere quattro pagine A6 per ogni pagina A4 finale, che deve essere piegata a metà, in orizzontale, per due volte, rilegando il tutto a segnature di sei fogli. In pratica, ogni segnatura si ottiene da tre fogli A4 che vengono piegati assieme per due volte.<br>Si ottiene un file con estensione '`.a6-4-a4-2h-6.ps`'. |
| `--a4-to-a6-4-a4-2h-8` | Prevede che l'argomento finale sia un file PostScript, in formato A4, che viene rielaborato in modo da ottenere quattro pagine A6 per ogni pagina A4 finale, che deve essere piegata a metà, in orizzontale, per due volte, rilegando il tutto a segnature di otto fogli. In pratica, ogni segnatura si ottiene da quattro fogli A4 che vengono piegati assieme per due volte.<br>Si ottiene un file con estensione '`.a6-4-a4-2h-8.ps`'. |
| `--a4-to-a6-4-a4-2h-10` | Prevede che l'argomento finale sia un file PostScript, in formato A4, che viene rielaborato in modo da ottenere quattro pagine A6 per ogni pagina A4 finale, che deve essere piegata a metà, in orizzontale, per due volte, rilegando il tutto a segnature di 10 fogli. In pratica, ogni segnatura si ottiene da cinque fogli A4 che vengono piegati assieme per due volte.<br>Si ottiene un file con estensione '`.a6-4-a4-2h-10.ps`'. |
| `--a4-to-a6-4-a4-1v-1` | Prevede che l'argomento finale sia un file PostScript, in formato A4, che viene rielaborato in modo da ottenere quattro pagine A6 per ogni pagina A4 finale, che deve essere piegata a metà, in verticale, rilegando il tutto a segnature di un foglio.<br>Si ottiene un file con estensione '`.a6-4-a4-1v-1.ps`'. |
| `--a7x4-to-a7x4-2-a4-1v-1` | Prevede che l'argomento finale sia un file PostScript, in formato A7x4, che viene rielaborato in modo da ottenere due pagine A7x4 per ogni pagina A4 finale, che deve essere piegata a metà, in verticale, rilegando il tutto a segnature di un foglio.<br>Si ottiene un file con estensione '`.a7x4-2-a4-1v-1.ps`'. |

| Opzione | Descrizione |
|---|---|
| `--a7x4-to-a7x4-2-a4-1v-10` | Prevede che l'argomento finale sia un file PostScript, in formato A7x4, che viene rielaborato in modo da ottenere due pagine A7x4 per ogni pagina A4 finale, che deve essere piegata a metà, in verticale, rilegando il tutto a segnature di 10 fogli.<br>Si ottiene un file con estensione '`.a7x4-2-a4-1v-10.ps`'. |

# Il documento secondo Alml

Il DTD di Alml è organizzato per gestire documenti molto grandi, che possono essere suddivisi in volumi, parti e capitoli. Tuttavia, la suddivisione in volumi o in parti resta facoltativa, mentre la divisione in capitoli è obbligatoria.

Alml non ha ancora raggiunto una sistemazione «definitiva» e si evolverà ancora assieme a *Appunti di informatica libera*. In questo capitolo non sono descritti tutti i dettagli sull'impostazione attuale del DTD di Alml; eventualmente si può studiare il DTD stesso. Tuttavia, il DTD non rappresenta in modo perfetto i vincoli che si pongono poi nella composizione.

Quando devono essere indicate delle dimensioni che prevedono la specificazione dell'unità di misura, si usano le sigle elencate nella tabella 10.1.

Tabella 10.1. Sigle delle unità di misura utilizzabili con Alml.

| Sigla | Unità di misura corrispondente |
|---|---|
| pt | Punti tipografici corrispondenti a 1/72,27 di pollice. |
| bp | Punti tipografici corrispondenti a 1/72 di pollice. |
| pc | Pica corrispondenti a 1/6 di pollice. |
| in | Pollici. |
| cm | Centimetri. |
| mm | Millimetri. |

## 10.1 Organizzazione generale

Secondo il DTD di Alml, il documento ha una struttura generale ben definita:

```
<!DOCTYPE ALML PUBLIC "-//D.G.//DTD Alml//EN">
<alml>
<head>
…
</head>
[<intro>
…
</intro>]
<body>
…
</body>
[<appendix>
…
</appendix>]
[<index>
…
</index>]
</alml>
```

In questa struttura, gli elementi '**`head`**' e '**`body`**' sono obbligatori, mentre gli altri possono essere omessi, se non sono richiesti.

Si può intuire il senso della cosa: l'elemento '**`head`**' serve a contenere informazioni amministrative, oltre a ciò che deve apparire nelle primissime pagine (il titolo dell'opera, il copyright ecc.); l'elemento '**`intro`**' permette di inserire dei capitoli speciali da trattare come introduzioni o prefazioni, che come tali non risultano numerate; l'elemento '**`body`**' permette di inserire capitoli, oppure parti, o volumi; l'elemento '**`appendix`**' permette di inserire capitoli da trattare come appendici, numerate convenzionalmente in modo letterale; infine, l'elemento '**`index`**' permette di inserire capitoli speciali per l'inclusione degli indici analitici.

## 10.2 Dalla copertina all'indice generale

L'elemento che delimita il documento nella sua interezza, '**alml**', può contenere due attributi facoltativi: '**lang**' e '**spacing**'. L'attributo '**lang**' permette di definire il linguaggio generale con cui è stato scritto il documento, attraverso una sigla secondo lo standard ISO 639.[1]

L'attributo '**spacing**' permette di definire il modo in cui vengono gestiti gli spazi alla fine dei periodi. Assegnando la parola chiave '**normal**', si ottiene la spaziatura normale della convenzione inglese, in cui lo spazio dopo un punto ha una lunghezza maggiore degli altri; in alternativa, assegnando la parola chiave '**uniform**', oppure '**french**', si ottiene una spaziatura uniforme, come richiede la tradizione tipografica italiana e anche di altri paesi.

In generale, un documento scritto in lingua italiana dovrebbe utilizzare l'elemento '**alml**' in questo modo:

```
<alml lang="it" spacing="uniform">
```

Tabella 10.2. Elementi SGML dalla copertina all'indice generale.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| alml | | Contenitore del documento. |
| *lang* | Attributo | Sigla ISO 639 del linguaggio. |
| *spacing* | Attributo | '**normal**', '**french**' e '**uniform**'. |
| head | | Intestazione del documento. |
| admin | | Informazioni amministrative. |
| description | | Descrizione in breve del documento. |
| keywords | | Elenco di parole chiave. |
| htmlmeta | | Contenuto di un elemento HTML '**META**'. |
| *name* | Attributo | Equivalente all'HTML. |
| *lang* | Attributo | Equivalente all'HTML. |
| chapterdefinition | | Definizione alternativa del capitolo. |
| partdefinition | | Definizione alternativa della parte. |
| tomedefinition | | Definizione alternativa del volume. |
| printedfontsize | | Corpo del carattere in punti. |
| *type* | Attributo | Definisce il contesto a cui si fa riferimento. |
| printedpagesize | | Margini e giustezza. |
| *type* | Attributo | Definisce il contesto a cui si fa riferimento. |
| title | %inline; | Titolo del documento. |
| subtitle | %inline; | Sottotitolo. |
| author | %inline; | Autore. |
| date | #PCDATA | Data. |
| edition | %inline; | Edizione, se diversa dalla data. |
| version | %inline; | Versione, se diversa dall'edizione. |
| frontcovertop | %block; | Blocco che precede il titolo. |
| abstract | %block; | Descrizione del contenuto. |
| frontcoverbottom | %block; | Testo aggiuntivo di copertina, dopo il titolo e le altre indicazioni standard. |
| backcover | %block; | Contenuto della copertina finale. |
| textbeforelegal | %block; | Testo prima delle informazioni legali. |
| legal | %block; | Informazioni legali. |
| dedications | %block; | Pagina della dedica. |
| textafterdedications | %block; | Testo successivo alla dedica. |
| maincontents | %block; | Inserimento dell'indice generale. |

[1] Quando le informazioni su un certo linguaggio non sono disponibili, si applicano le convenzioni inglesi.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| *levels* | Attributo | Livelli di dettaglio dell'indice. |
| *nopages* | Attributo | '**true**', '**false**'. |

La tabella 10.2 mostra in breve l'elenco degli elementi che riguardano l'intestazione del documento; cosa che contiene tutte le informazioni per realizzare la copertina, fino ad arrivare all'indice generale.

Si può osservare che tutto è contenuto nell'elemento '**head**', all'inizio del quale prende posto un altro «contenitore» denominato '**admin**'. Al suo interno sono previsti elementi relativi a informazioni amministrative, in particolare '**description**' e '**keywords**', il cui scopo è quello di generare degli elementi '**META**' corrispondenti nella composizione HTML:

```
<HEAD>
    ...
    <META NAME="Description" CONTENT="An example for Alml documentation system">
    <META NAME="Keywords" CONTENT="SGML, XML, HTML, Alml">
    ...
</HEAD>
```

Inoltre, si possono aggiungere anche altri elementi '**META**' di HTML, attraverso l'elemento '**HTMLMETA**', come si vede nell'esempio seguente:

```
<head>
    <admin>
        <description>GNU/Linux e altro software libero</description>

        <keywords>Linux, GNU/Linux, Unix, software, software libero,
        free software</keywords>

        <htmlmeta name="Resource-type" lang="en">Document</htmlmeta>
        <htmlmeta name="Revisit-after" lang="en">15 days</htmlmeta>
        <htmlmeta name="Robots">ALL</htmlmeta>
    </admin>
    ...
    ...
</HEAD>
```

Gli elementi '**chapterdefinition**', '**partdefinition**' e '**tomedefinition**' vengono descritti più avanti in questo capitolo (sezione 10.8).

L'elemento '**printedfontsize**' consente di definire l'altezza del carattere indicato attraverso l'attributo '**type**', per la composizione stampata.

L'elemento '**printedpagesize**' consente di definire I margini e la giustezza per la composizione stampata, in base al contesto indicato dall'attributo '**type**'.

L'elemento '**title**' serve a indicare il titolo del documento; gli elementi eventuali '**subtitle**' permettono di inserire dei sottotitoli successivi.

L'elemento '**abstract**', facoltativo, permette l'inserimento di una descrizione, più o meno articolata, composta da blocchi di testo (ciò che nella tabella viene rappresentato schematicamente dalla macro '**%block;**').

Successivamente è possibile inserire uno o più elementi '**author**', uno per il nominativo di ogni coautore, eventualmente.

Gli elementi '**date**' e '**edition**' servono per indicare una data o una sigla differente che rappresenti in qualche modo l'edizione. In generale dovrebbe essere sufficiente l'indicazione di uno solo di questi due elementi.

L'elemento '**frontcovertop**' permette l'inserzione di blocchi prima del titolo; così, l'elemento '**frontcoverbottom**' consente di fare la stessa cosa dopo il titolo e le altre indicazioni standard. L'elemento '**backcover**' permette di definire il contenuto della copertina finale.

Gli elementi successivi riguardano la seconda pagina assoluta e quelle successive.

Nella seconda pagina appaiono di solito le informazioni sul copyright, nella parte bassa, mentre nella parte superiore potrebbero esserci altre informazioni, come una breve descrizione degli autori. L'elemento '**textbeforelegal**' permette di inserire blocchi di testo da collocare nella prima parte della seconda pagina, mentre l'elemento '**legal**' è fatto per le informazioni legali, a partire dal copyright.

Dopo le informazioni legali è possibile inserire una pagina di dediche, attraverso l'elemento '**dedications**'. Eventualmente, se necessario, è possibile aggiungere altre notizie all'interno dell'elemento '**textafterdedications**' che segue le dediche.

Infine, è possibile collocare l'elemento vuoto '**maincontents**' per ottenere l'inserimento dell'indice generale. L'attributo '**levels**' permette di definire il livello di dettaglio desiderato dell'indice: il numero zero rappresenta il minimo e fa in modo di ottenere informazioni fino alle parti, mentre valori superiori aumentano il dettaglio. Assegnando all'attributo '**nopages**' il valore '**true**', si richiedere espressamente l'eliminazione dei riferimenti ai numeri di pagina; cosa che può essere utile soltanto nella composizione per la stampa.

### 10.2.1 Margini e giustezza nella composizione stampata

È possibile definire i margini e la giustezza (la larghezza del testo) della composizione stampata, senza dover intervenire modificando lo stile TeX. Si utilizza per questo l'elemento '**printedpagesize**', all'interno dell'elemento '**admin**', nell'intestazione del documento, specificando il contesto con l'attributo '**type**'. Si osservi l'esempio in cui si mostrano tutti i valori disponibili per l'attributo '**type**':

```
<head>
    <admin>
        ...
        <printedpagesize type="topmargin">2.5cm</printedpagesize>
        <printedpagesize type="bottommargin">2.5cm</printedpagesize>
        <printedpagesize type="internalmargin">3.5cm</printedpagesize>
        <printedpagesize type="bodywidth">15cm</printedpagesize>
        ...
    </admin>
...
</head>
```

Il tipo '**topmargin**' è il margine superiore, fino alla **base** del testo normale (la riga di intestazione viene collocata automaticamente); il tipo '**bottommargin**' è il margine inferiore; il tipo '**internalmargin**' è il margine sinistro per le pagine destre e il margine destro per le pagine sinistre; il tipo '**bodywidth**' è la giustezza, ovvero la larghezza della colonna in cui scorre effettivamente il testo.

Quando la composizione non richiede un margine sinistro diverso da quello destro, si ignora il valore del margine interno, utilizzando margini uguali.

Tabella 10.3. Valori dell'attributo '**type**' dell'elemento '**printedpagesize**'.

| Valore | Contesto a cui si fa riferimento |
|---|---|
| topmargin | Margine superiore. |
| bottommargin | Margine inferiore. |
| internalmargin | Margine interno. |
| bodywidth | Giustezza. |

Il margine esterno non viene indicato, perché si preferisce indicare la giustezza, essendo un valore che è meglio non vari automaticamente, dal momento che da questo dipendono anche le dimensioni che si assegnano ad altri componenti contenuti nel testo.

Il formato della carta viene definito al di fuori del sorgente SGML, attraverso le opzioni di Alml. Ciò permette di produrre composizioni differenti a seconda del tipo di carta disponibile. Tuttavia, è evidente che le dimensioni adottate per la carta devono essere compatibili con i margini e la giustezza richiesti nel sorgente SGML.

### 10.2.2 Corpo del carattere nella composizione stampata

È possibile definire il corpo del carattere, nella composizione stampata, in alcune situazioni importanti, senza dover intervenire modificando lo stile TeX. Si utilizza per questo l'elemento '**`printedfontsize`**', all'interno dell'elemento '**`admin`**', nell'intestazione del documento, specificando il contesto con l'attributo '**`type`**'. Si osservi l'esempio:

```
<head>
    <admin>
        ...
        <printedfontsize type="normal">4mm</printedfontsize>
        <printedfontsize type="table">3.5mm</printedfontsize>
        ...
    </admin>
...
</head>
```

Il tipo '**`normal`**' è il carattere normale del testo; il tipo '**`table`**' è il carattere utilizzato nelle tabelle di Alml. La distanza tra le righe viene impostata automaticamente al 120 % della dimensione del carattere utilizzato.

La dimensione del carattere deve essere armoniosa rispetto al resto del documento. Bisogna provare per rendersi conto se il risultato che si ottiene è accettabile oppure no.

Tabella 10.4. Valori dell'attributo '**`type`**' dell'elemento '**`printedfontsize`**'.

| Valore | Carattere a cui si fa riferimento |
|---|---|
| title | Titolo dell'opera che appare in copertina. |
| tomeheading | Titolo dei volumi nella loro pagina iniziale. |
| h0 | Titolo delle parti nella loro pagina iniziale. |
| h1 | Titolo dei capitoli. |
| h2 | Titolo delle sezioni di primo livello. |
| h3 | Titolo delle sottosezioni. |
| h4 | Titolo delle sotto-sottosezioni. |
| normal | Testo normale. |
| table | Testo delle tabelle. |

## 10.3 Contenuto

Il contenuto del documento si articola in tre blocchi fondamentali: '**intro**', '**body**' e '**appendix**'. In coda, possono apparire degli indici analitici, racchiusi nel blocco dell'elemento '**index**'.

Questa classificazione in blocchi va a compensare la mancanza di elementi atti a circoscrivere l'estensione delle sezioni in cui si articola il testo. La mancanza di una strutturazione dettagliata delle sezioni[2] fa sì che in presenza di errori di sintassi SGML, l'analizzatore tenda a segnalare in seguito una quantità di errori inesistenti che non vanno considerati. In tali situazioni, si correggono i primi errori evidenti e si ripete la verifica SGML.

### 10.3.1 Introduzione

Dopo l'elemento '**head**' è prevista la possibilità di inserire l'elemento '**intro**', il cui scopo è quello di delimitare uno o più capitoli speciali, da intendere come prefazioni o introduzioni a vario titolo.

Per la definizione del capitolo, si veda quanto descritto a proposito dell'elemento '**body**'.

### 10.3.2 Corpo

Il corpo vero e proprio del documento è contenuto nell'elemento '**body**', il quale si può articolare in volumi, parti o capitoli. Sta all'autore scegliere quale livello di suddivisione superiore adottare. È evidente che se si usa una suddivisione in volumi, si prevede una sottoclassificazione in parti, che poi si dividono in capitoli; se si usa una suddivisione in parti, è obbligatoria una sottoclassificazione in capitoli.

> Eccezionalmente, un volume può contenere solo capitoli, senza parti, quando per qualche ragione ciò è necessario.

Volumi, parti, capitoli e sezioni inferiori sono delimitate materialmente attraverso la dichiarazione del titolo relativo, come avviene in HTML. La tabella 10.5 elenca gli elementi relativi, assieme agli attributi eventuali.

> Si osservi che nella composizione stampata, non è garantito che il titolo di una sezione si trovi nella stessa pagina in cui inizia il testo della sezione stessa.

Tabella 10.5. Dichiarazione dei titoli di volumi, parti, capitoli e sezioni inferiori, oltre ad altri elementi essenziali nella definizione della scomposizione del testo.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| tomeheading | %inline; | Titolo del volume. |
| *id* | Attributo | Ancora di riferimento. |
| *lang* | Attributo | Linguaggio del volume. |
| *bookmark* | Attributo | Testo da usare come segnalibro alternativo per la composizione PDF. |
| tomecontents | Vuoto | Indice generale del volume. |

[2] Qui si intendono sezioni a qualsiasi livello, compresi i capitoli, le parti e i volumi.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| *levels* | Attributo | Livello di dettaglio dell'indice. |
| *nopages* | Attributo | '**true**', '**false**'. |
| h0 | %inline; | Titolo della parte. |
| *id* | Attributo | Ancora di riferimento. |
| *lang* | Attributo | Linguaggio della parte. |
| *bookmark* | Attributo | Testo da usare come segnalibro alternativo per la composizione PDF. |
| partcontents | Vuoto | Indice generale della parte. |
| *levels* | Attributo | Livello di dettaglio dell'indice. |
| *nopages* | Attributo | '**true**', '**false**'. |
| h1 | %inline; | Titolo del capitolo. |
| *id* | Attributo | Ancora di riferimento. |
| *lang* | Attributo | Linguaggio del capitolo. |
| *bookmark* | Attributo | Testo da usare come segnalibro alternativo per la composizione PDF. |
| chaptercontents | Vuoto | Indice generale del capitolo. |
| *levels* | Attributo | Livello di dettaglio dell'indice. |
| *nopages* | Attributo | '**true**', '**false**'. |
| h2 | %inline; | Titolo della sezione. |
| *id* | Attributo | Ancora di riferimento. |
| *bookmark* | Attributo | Testo da usare come segnalibro alternativo per la composizione PDF. |
| h3 | %inline; | Titolo della sottosezione. |
| *id* | Attributo | Ancora di riferimento. |
| *bookmark* | Attributo | Testo da usare come segnalibro alternativo per la composizione PDF. |
| h4 | %inline; | Titolo della sotto-sottosezione. |
| *id* | Attributo | Ancora di riferimento. |
| *bookmark* | Attributo | Testo da usare come segnalibro alternativo per la composizione PDF. |
| segment | | Segmento di testo finale. |
| segmenthead | %inline; | Titolo di un segmento. |
| extramaincontents | Vuoto | Indice generale dell'opera, piazzabile ovunque. |
| *levels* | Attributo | Livello di dettaglio dell'indice. |
| *nopages* | Attributo | '**true**', '**false**'. |
| endofchapter | %inline; | Riga finale del capitolo. |

Nella parte iniziale delle classificazioni principali (volumi, parti e capitoli), è possibile collocare la richiesta di inserimento di un indice generale specifico. Si ottiene questo con gli elementi: '**tomecontents**', '**partcontents**' e '**chaptercontents**' (è disponibile anche l'elemento '**extramaincontents**' che riguarda l'opera intera e può essere collocato ovunque). Ognuno di questi elementi prevede l'attributo '**levels**', con il quale è possibile stabilire il livello di dettaglio di tali indici, tenendo presente che con il numero zero si ottengono voci fino alle parti, con uno si ottengono anche i capitoli, mentre con valori superiori si accede alle sezioni di livello inferiore. Anche in questo caso è possibile inibire la segnalazione delle pagine (nel caso di composizione per la stampa), utilizzando l'attributo '**nopages**'.

Alla fine del testo di ognuna di queste classificazioni, prima dell'inizio di una sottoclassificazione eventuale, è possibile collocare un «segmento» di testo, con un titolo che assomiglia a una voce di un elenco descrittivo. Si tratta dell'elemento '**segment**', i cui titoli si indicano nell'elemento '**segmenthead**'. Questo gruppo rappresenta un'anomalia nell'organizzazione generale,

introdotta solo per mantenere la compatibilità con le convenzioni usate in passato nella redazione di questa opera e in seguito potrebbe essere eliminato.

Infine, sempre per mantenere la compatibilità con il passato, esiste l'elemento '**endofchapter**', il cui scopo è quello di consentire l'inserimento di una riga di informazioni alla fine del capitolo.

### 10.3.3 Appendici

Dopo il corpo è possibile inserire l'elemento '**appendix**', il cui scopo è quello di delimitare uno o più capitoli speciali, da intendere come appendici.

### 10.3.4 Indici analitici

Alml consente la definizione di diversi tipi di indici analitici. Per questi è previsto uno spazio speciale collocato dopo le appendici, se ci sono, o in caso contrario subito dopo il corpo. Si tratta dell'elemento '**index**', che prevede l'inserimento di capitoli, come nel caso delle appendici.

L'inserimento di un elenco riferito a un indice analitico particolare si ottiene con l'elemento vuoto '**printindex**'. Verrà descritto meglio in seguito l'uso di questo elemento, perché Alml è in grado di gestire più indici analitici differenti.

### 10.3.5 Suddivisioni speciali

Oltre alle suddivisioni standard nella forma '**h*n***', sono disponibili altre suddivisioni per scopi particolari. Sono previsti due capitoli speciali per le presentazioni (diapositive o lucidi per lavagna luminosa) e per i prospetti schematici riassuntivi (tavole sintetiche e simili), oltre a due tipi di sezioni per domande e risposte.

Tabella 10.6. Dichiarazione dei titoli di capitoli e sezioni speciali.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| slideh1 | %inline; | Titolo della diapositiva o del lucido. |
| *id* | Attributo | Ancora di riferimento. |
| *lang* | Attributo | Linguaggio della diapositiva o del lucido. |
| sheeth1 | %inline; | Titolo della scheda sintetica riassuntiva. |
| *id* | Attributo | Ancora di riferimento. |
| *lang* | Attributo | Linguaggio della scheda sintetica. |
| faqh2 | %inline; | Titolo del gruppo di domande e risposte. |
| *id* | Attributo | Ancora di riferimento. |
| faqh3 | %inline; | Domanda a cui segue una risposta. |
| *id* | Attributo | Ancora di riferimento. |

Osservando la tabella 10.6, si può intuire che gli elementi '**slideh1**' e '**sheeth1**' si usano al posto di un capitolo normale. La differenze più importante rispetto all'elemento '**h1**', sta nel fatto che non possono contenere altre suddivisioni in sezioni; inoltre, nella composizione per la stampa non appare il numero della pagina. Anche se non c'è modo di controllare la dimensione del contenuto, è bene che ogni diapositiva e ogni scheda occupi una sola pagina nella composizione per la stampa.

L'elemento '**faqh2**' va usato al posto di '**h2**', all'interno di un capitolo normale. Permette di introdurre un gruppo di domande e risposte, precedendole eventualmente da qualche blocco di testo introduttivo.

L'elemento '**faqh3**' serve a contenere il testo di una domanda, anche se potrebbe essere più lungo di un titolo normale. Il testo viene rappresentato in modo evidenziato, ma non tanto quanto un elemento '**h3**' normale. Dopo l'elemento '**faqh3**' ci si aspetta di trovare la risposta alla domanda.

Eventualmente, la struttura composta da '**faqh2**' e '**faqh3**' può essere utilizzata anche per realizzare dei questionari o dei test valutativi.

## 10.4 Documento multilingua

Oltre a indicare il linguaggio nell'elemento '**alml**', attraverso l'attributo '**lang**', lo stesso attributo è disponibile all'inizio dei volumi, delle parti e dei capitoli. In pratica, si può usare l'attributo '**lang**' anche negli elementi '**tomeheading**', '**h0**' e '**h1**'.

Contrariamente alla logica comune, in questo caso l'attributo '**lang**' attribuisce il valore della scelta linguistica a tutto il volume, alla parte o al capitolo relativo. Un volume, una parte o un capitolo che non abbiano la definizione esplicita di un linguaggio, ereditano la definizione del livello precedente.

La motivazione più importante per la quale è stato introdotto questo attributo nella dichiarazione dei volumi, delle parti e dei capitoli, sta nel fatto che così la composizione in HTML genera file con intestazioni adeguate, anche per l'indicizzazione delle informazioni.

La sigla della lingua va attribuita secondo lo standard ISO 639. Se non è stata prevista la traduzione dei termini relativi alla composizione nella lingua richiesta, questi si ottengono in inglese.

L'esempio seguente mostra la dichiarazione esplicita di un capitolo che è da considerare in lingua inglese:

```
<h1 lang="en">Here I am</h1>
```

La definizione del volume, della parte o del capitolo viene adattata alla lingua, solo se questa non è stata modificata attraverso gli elementi '**tomedefinition**', '**partdefinition**' e '**chapterdefinition**', descritti più avanti in questo capitolo.

## 10.5 Blocchi di testo ed elementi inseriti all'interno delle righe

A parte gli elementi strutturali del documento, il DTD di Alml organizza il testo in due gruppi fondamentali: i blocchi di testo, a cui corrisponde l'entità parametrica '**%block;**', e gli elementi collocabili all'interno delle righe, corrispondente all'entità '**%inline;**'. Il caso tipico di elemento che costituisce un blocco di testo è il paragrafo, '**p**', mentre il caso tipico di elemento che costituisce un'inserzione nella riga è l'enfatizzazione, '**em**'. La tabella 10.7 riepiloga gli elementi comuni che riguardano inserzioni all'interno della riga, mentre quelli che rappresentano un blocco e altri elementi speciali sono descritti separatamente in sezioni apposite.

Tabella 10.7. Elementi inseriti all'interno delle righe.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| em | %inline; | Enfasi normale. |
| strong | %inline; | Enfasi rafforzata. |
| big | %inline; | Testo relativamente più grande. |
| small | %inline; | Testo relativamente più piccolo. |
| acronym | %inline; | Acronimo. |
| dacronym | %inline; | Descrizione di un acronimo. |
| kbd | %inline; | Tasto. |
| button | %inline; | Bottone o tasto grafico. |
| menuitem | %inline; | Voce di un menù. |
| asciicode | %inline; | Codice ASCII. |
| code | %inline; | Codice (come in HTML). |
| samp | %inline; | Stringa (come in HTML). |
| kerneloption | %inline; | Opzione del kernel. |
| file | var \| em \| #PCDATA | File o directory. |
| dfn | #PCDATA \| special | Definizione. |
| strdfn | %inline; | Definizione in lingua straniera. |
| special | #PCDATA | Termine speciale per qualche ragione. |
| *special* | Attributo | Nome attribuito al genere del termine. |
| sup | var \| em \| strong \| #PCDATA | Apice. |
| sub | var \| em \| strong \| #PCDATA | Pedice. |
| pwr | var \| em \| strong \| #PCDATA | Potenza (esponente). |
| navlink | #PCDATA | Riferimento per la navigazione HTML. |

### 10.5.1 Numeri

La rappresentazione uniforme di valori numerici, specie quando si opera spesso con basi di numerazione insolite, diventa un aspetto delicato. Alml prevede alcuni elementi da utilizzare all'interno delle righe per delimitare valori numerici, eventualmente con basi di numerazioni particolari, come si vede nella tabella 10.8.

Tabella 10.8. Elementi inseriti all'interno delle righe per la rappresentazione uniforme di valori numerici.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| num | [+-]?[0-9]+[.,]?[0-9]* | Numero decimale comune. |
| exa | var \| em \| strong \| #PCDATA | Numero in base 16. |
| dec | var \| em \| strong \| #PCDATA | Numero in base 10. |
| oct | var \| em \| strong \| #PCDATA | Numero in base 8. |
| bin | var \| em \| strong \| #PCDATA | Numero in base 2. |

Il caso dell'elemento '**num**' è speciale. In particolare, si fa riferimento a un numero in base 10, in cui non si mostra la base di numerazione, ma si usa una modalità di rappresentazione standard. Per questa ragione, il numero in questione deve essere inserito come previsto, utilizzando la virgola o il punto come separatore della parte decimale,[3] aggiungendo il segno all'inizio, se

[3] Il segno meno, va indicato con il trattino normale.

necessario, senza usare altri spazi o altri caratteri. Il numero viene elaborato separando le cifre a terne.

Per quanto riguarda gli altri elementi, a seconda del tipo di composizione si utilizza un modo diverso per mostrare la base di numerazione. Tuttavia, in questi casi il contenuto degli elementi non è strettamente letterale, come si vede dalla tabella.

### 10.5.2 Elenchi e simili

Gli elenchi di Alml sono molto semplici. Si tratta dei soliti elenchi puntati, numerati e descrittivi. Questi si comportano in modo molto simile all'HTML; la differenza sostanziale sta nel fatto che il contenuto delle voci è composto da uno o più blocchi di testo, mentre in HTML è consentita anche la presenza di righe pure e semplici.

Tabella 10.9. Elenchi.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| dl | | Elenco descrittivo. |
| dt | %inline; | Termine descrittivo. |
| dd | %block; | Descrizione relativa. |
| ol | | Elenco numerato. |
| li | %block; | Elemento dell'elenco. |
| ul | | Elenco puntato. |
| li | %block; | Elemento dell'elenco. |

### 10.5.3 Testo letterale o quasi

L'inclusione di testo letterale in un sorgente SGML è sempre un problema. Alml prevede tre ambienti diversi: '**`verbatimpre`**', '**`asciiart`**' e '**`pre`**'. Nei primo due casi si può scrivere senza alcuna preoccupazione, tranne per il fatto che il testo va inserito in una sezione marcata di tipo '**`CDATA`**'; nel terzo caso invece, è necessario comportarsi come nel testo normale, utilizzando le entità standard quando servono, potendo includere anche gran parte degli elementi che rappresentano un'inserzione all'interno di una riga. In entrambi i casi vengono rispettate le interruzioni di riga.

```
<verbatimpre>
<![CDATA[
uno
   &
    due
]]>
</verbatimpre>
```

```
<pre>
uno
   &amp;
    due
</pre>
```

I due esempi portano allo stesso risultato:

```
uno
   &
    due
```

In generale si sceglierà il primo o il secondo modo (quando appropriato), mentre il terzo lo si riserva ai casi in cui si devono inserire le cose che i primi due non possono contenere.

Gli elementi '**verbatimpre**' e '**pre**' possono anche essere bordati e numerati. L'esempio seguente mostra l'uso dell'elemento '**verbatimpre**', dove le righe del suo contenuto devono essere numerate a partire dal numero uno:

```
<verbatimpre numbering="1">
<![CDATA[
drwxr-xr-x    2 root     root         4096 2003-01-17 15:47 bin
drwxr-xr-x    3 root     root         4096 2003-01-28 16:18 boot
drwxr-xr-x    1 root     root            0 1970-01-01 01:00 dev
drwxr-xr-x  139 root     root         8192 2003-01-30 16:47 etc
drwxrwsr-x   17 root     staff        4096 2003-01-19 22:01 home
drwxr-xr-x    6 root     root         4096 2003-01-11 15:26 lib
drwxr-xr-x    2 root     root        16384 2000-12-15 14:49 lost+found
drwxr-xr-x  311 root     root         8192 2003-01-22 16:36 mnt
dr-xr-xr-x   89 root     root            0 2003-01-30 14:30 proc
drwxr-xr-x   15 root     root         4096 2003-01-30 16:32 root
drwxr-xr-x    2 root     root         4096 2003-01-10 16:04 sbin
drwxrwxrwt    5 root     root       176128 2003-01-30 17:45 tmp
drwxr-xr-x   15 root     root         4096 2003-01-04 11:06 usr
drwxr-xr-x   16 root     root         4096 2002-10-27 18:25 var
]]>
</verbatimpre>
```

Ecco cosa si ottiene:

```
1     drwxr-xr-x    2 root     root         4096 2003-01-17 15:47 bin
2     drwxr-xr-x    3 root     root         4096 2003-01-28 16:18 boot
3     drwxr-xr-x    1 root     root            0 1970-01-01 01:00 dev
4     drwxr-xr-x  139 root     root         8192 2003-01-30 16:47 etc
5     drwxrwsr-x   17 root     staff        4096 2003-01-19 22:01 home
6     drwxr-xr-x    6 root     root         4096 2003-01-11 15:26 lib
7     drwxr-xr-x    2 root     root        16384 2000-12-15 14:49 lost+found
8     drwxr-xr-x  311 root     root         8192 2003-01-22 16:36 mnt
9     dr-xr-xr-x   89 root     root            0 2003-01-30 14:30 proc
10    drwxr-xr-x   15 root     root         4096 2003-01-30 16:32 root
11    drwxr-xr-x    2 root     root         4096 2003-01-10 16:04 sbin
12    drwxrwxrwt    5 root     root       176128 2003-01-30 17:45 tmp
13    drwxr-xr-x   15 root     root         4096 2003-01-04 11:06 usr
14    drwxr-xr-x   16 root     root         4096 2002-10-27 18:25 var
```

L'esempio seguente mostra l'uso dell'elemento '**pre**', bordato:

```
<pre border="1">
uno
   &amp;
    due
</pre>
```

Ecco il risultato:

```
uno
   &
    due
```

In un documento a carattere tecnico-informatico, è essenziale la possibilità di indicare dei modelli sintattici. Alml prevede l'uso di un elemento simile a '**pre**', dedicato precisamente a questo scopo: '**syntax**'.

```
<syntax>
man <synsqb><var>n_sezione</var></synsqb> <var>nome</var>
</syntax>
```

All'interno di questo elemento si possono inserire altri elementi specifici per rappresentare i componenti della sintassi. Infatti, è necessario distinguere tra parole chiave, metavariabili e altre indicazioni. In generale, quello che si scrive normalmente deve essere inteso come un dato fisso, ovvero delle parole chiave o delle stringhe fisse. Per indicare un contenuto variabile si utilizza l'elemento '**var**' per delimitare la denominazione di un qualcosa di variabile (un'opzione o simile).

Altri elementi speciali servono a guidare la lettura della sintassi: '**synsqb**' delimita una parte della sintassi che va intesa come facoltativa e si traduce generalmente con delle parentesi quadre che, se possibile, si distinguono dal testo normale; '**syncub**' delimita una parte della sintassi che va intesa come un corpo unico e si traduce generalmente con delle parentesi graffe speciali; '**synverbar**' (elemento vuoto) indica un'alternativa e si rappresenta con una barra verticale; '**synellipsis**' (elemento vuoto) rappresenta dei puntini di sospensione particolari, diversi da quelli che si otterrebbero in modo normale. Nell'uso di questi elementi occorre sempre un po' di prudenza, tenendo conto dei tipi di composizione in cui non è possibile mostrare questi simboli in forme diverse dal normale.

Tabella 10.10. Elementi SGML che riguardano la rappresentazione di testo preformattato.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| pre | %inline; | Testo preformattato. |
| *width* | Attributo | Ampiezza massima in caratteri del testo; zero richiede espressamente le stesse dimensioni del contesto. |
| *border* | Attributo | Si può attribuire solo il valore zero o il valore uno. Con il valore uno si ottiene l'aggiunta di un bordo. |
| *numbering* | Attributo | Si può attribuire la stringa nulla o un numero intero, con o senza segno: se si attribuisce un valore numerico si ottiene la numerazione delle righe contenute a partire da quel valore. |
| pnewline | Vuoto | Continuazione nella riga successiva. |
| verbatimpre | testo letterale | Testo letterale preformattato. |
| *width* | Attributo | Ampiezza massima in caratteri del testo; zero richiede espressamente le stesse dimensioni del contesto. |
| *border* | Attributo | Si può attribuire solo il valore zero o il valore uno. Con il valore uno si ottiene l'aggiunta di un bordo. |
| *numbering* | Attributo | Si può attribuire la stringa nulla o un numero intero, con o senza segno: se si attribuisce un valore numerico si ottiene la numerazione delle righe contenute a partire da quel valore. |
| asciiart | testo letterale | Testo letterale preformattato. |
| *width* | Attributo | Ampiezza massima in caratteri del testo; zero richiede espressamente le stesse dimensioni del contesto. |
| syntax | %inline; | Modello sintattico preformattato. |

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| *width* | Attributo | Ampiezza massima in caratteri del testo; zero richiede espressamente le stesse dimensioni del contesto. |
| *sep* | Attributo | '**none**', '**border**'. |
| synsqb | %inline; | Parentesi quadre di un modello sintattico. |
| syncub | %inline; | Parentesi graffe di un modello sintattico. |
| synverbar | %inline; | Barra verticale di un modello sintattico. |
| var | %inline; | Metavariabile sintattica. |
| synellipsis | Vuoto | Ellissi nei modelli sintattici. |
| snewline | Vuoto | Continuazione nella riga successiva. |

Si tenga in considerazione il fatto che gli elementi '**synsqb**', '**syncub**', '**synverbar**', '**synellipsis**' e '**var**', possono essere utilizzati anche al di fuori dell'elemento '**syntax**', in qualità di inserzioni normali nelle righe.

La riga di un modello sintattico che si estende troppo in orizzontale, può essere spezzata e ripresa inserendo l'elemento vuoto '**snewline**', in modo da ottenere una segnalazione evidente nella composizione finale, senza lasciare ambiguità. La stessa cosa, eventualmente, si può fare nell'elemento '**pre**', usando l'elemento vuoto '**pnewline**'. Si osservi l'esempio seguente che si riferisce a un modello sintattico:

```
<syntax sep="border">
pippo --primo <synverbar> <snewline>--secondo <synverbar> --terzo
</syntax>
```

Ecco cosa si ottiene:

```
pippo --primo | ↩
↪--secondo | --terzo
```

Quando si usa un elemento '**snewline**', '**pnewline**' o '**cnewline**', vicino a uno spazio orizzontale, è bene che lo spazio venga lasciato prima dell'inserzione dell'elemento stesso, senza eliminarlo, in modo da sottolinearne la presenza.

Gli elementi '**pre**', '**verbatimpre**', '**asciiart**' e '**syntax**', sono predisposti inizialmente per poter rappresentare 80 colonne di testo letterale, in una larghezza pari a quella normale del testo. In situazioni particolari può essere necessario ampliare o ridurre la dimensione dei caratteri nella composizione stampata, per consentire la rappresentazione di un testo più ampio orizzontalmente. In questi casi, si può utilizzare l'attributo '**width**', assegnando la quantità di colonne che si desiderano.

Viene lasciata la facoltà di ridefinire la larghezza del testo anche negli elementi '**syntax**' e '**pre**', benché sia possibile interrompere e riprendere le righe troppo lunghe, come già descritto, con gli elementi '**snewline**' e '**pnewline**'.

```
<pre width="90">
1234567890
          1234567890
                    1234567890
                              1234567890
                                        1234567890
                                                  1234567890
                                                            1234567890
                                                                      1234567890
                                                                                1234567890
</pre>
```

L'esempio mostra l'utilizzo di un elemento '**pre**' per rappresentare un testo che si estende su 90 colonne.

In caso di necessità, se si vuole che la dimensione del carattere sia la stessa dell'ambiente in cui si trova l'elemento in questione, è sufficiente richiedere espressamente una larghezza pari a zero.

### 10.5.4 Comandi

I comandi che si impartiscono attraverso una riga di comando, possono essere rappresentati con l'elemento '**command**'. Si osservi l'esempio seguente:

```
<command><prompt>$ </prompt><type>ls</type><kbd>Invio</kbd></command>
```

Nell'ambito dell'elemento '**command**' è quasi tutto facoltativo; tuttavia, l'invito, rappresentato dall'elemento '**prompt**', va messo per primo. Dopo l'elemento '**type**', che serve a delimitare il testo che viene inserito sulla riga di comando, è possibile anche specificare il tasto che serve a concludere la digitazione, come in questo caso, oppure se ne può fare a meno, lasciandolo sottinteso.

Il testo che viene restituito da un comando si rappresenta normalmente con l'elemento '**verbatimpre**'.

A volte, si ha la necessità di rappresentare dei comandi piuttosto lunghi, che nella composizione stampata potrebbero risultare spezzati in modo imprevedibile e indesiderabile. È possibile indicare esplicitamente dove spezzare il comando, facendo in modo che nella composizione si intenda chiaramente questo fatto. Per questo si usa l'elemento vuoto '**cnewline**', che si inserisce all'interno di '**type**'.

Tabella 10.11. Elementi SGML che servono a rappresentare un comando.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| command | | Comando da digitare. |
| prompt | %inline; | Stringa dell'invito. |
| type | %inline; | Digitazione del comando. |
| cnewline | Vuoto | Continua il comando a riga nuova. |
| kbd | %inline; | Tasto o combinazione di tasti. |
| button | %inline; | Bottone o tasto grafico. |

### 10.5.5 Figure

Alml permette di gestire le figure in diversi modi. In generale può trattarsi di file di immagini, oppure di altre cose, come dei disegni ASCII racchiusi nell'elemento '**verbatimpre**' o in '**asciiart**'.

L'ambiente normale in cui si inserisce una figura è quello dato dall'elemento '**figure**', che in particolare può essere definito come fluttuante oppure fisso nel punto in cui si trova. All'interno di questo elemento può essere collocata una figura costituita da un'immagine esterna, da un'immagine incorporata, oppure un blocco di testo normale, come un elemento '**verbatimpre**' per realizzare un disegno ASCII.

```
<figure id="f-esempio-1">
<fcaption>
    Figura <figureref>. Ecco il mio primo esempio.
</fcaption>
<image imgfile="esempio-1" height="4cm">
</figure>
```

L'esempio mostra la situazione più comune. Si tratta dell'incorporazione del file '`esempio-1.png`', dove viene stabilita l'altezza di quattro centimetri, lasciando che la larghezza si adatti di conseguenza, in modo relativo. Si può osservare che l'elemento '**figure**' contiene un attributo '**id**', con lo scopo evidente di potervi fare riferimento.

L'elemento '**fcaption**' serve a delimitare il testo che si vuole fare apparire come didascalia. Al suo interno si nota la presenza di un elemento vuoto, '**figureref**', che in questo caso rappresenta un riferimento all'ultima figura, cioè a se stessa.

Una figura ASCII potrebbe essere realizzata, per esempio, nel modo seguente, come in tanti altri modi possibili che fanno uso di blocchi di testo:

```
<figure id="f-esempio-1">
<fcaption>
    Figura <figureref>. Ecco il mio primo esempio.
</fcaption>
<pre>
    pinco &amp; pallino
          |
          '--&gt; e-commerciale
</pre>
</figure>
```

Oltre all'elemento '**figure**', esiste l'elemento '**img**' per le immagini inserite nel testo.

```
<p>Bla bla bla <img imgfile="f-esempio-1" alt="Esempio" height="4mm"> bla
bla bla.</p>
```

Tabella 10.12. Elementi SGML che servono a rappresentare delle figure di qualche tipo.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| figure | | Involucro di una figura normale. |
| *id* | Attributo | Ancora di riferimento per la figura. |
| *pos* | Attributo | '**fixed**', '**float**'. |
| *sep* | Attributo | '**none**', '**rule**', '**border**'. |
| fcaption | %inline; | Didascalia. |
| asciiart | testo letterale | Codice ASCII letterale preformattato. |

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| *width* | Attributo | Numero di colonne, in caratteri, del testo. |
| image | Vuoto | Riferimento a un'immagine esterna. |
| *imgfile* | Attributo | File contenente l'immagine, senza estensione. |
| *height* | Attributo | Altezza dell'immagine. |
| *width* | Attributo | Larghezza dell'immagine. |
| embimage | #PCDATA | Immagine incorporata; algoritmo Base64. |
| *alt* | Attributo | Descrizione alternativa alla visualizzazione. |
| *height* | Attributo | Altezza dell'immagine. |
| *width* | Attributo | Larghezza dell'immagine. |
| epsimage | #PCDATA | Codice EPS letterale. |
| *alt* | Attributo | Descrizione alternativa alla visualizzazione. |
| *width* | Attributo | Larghezza. |
| *height* | Attributo | Altezza. |
| figimage | #PCDATA | Codice XFig letterale. |
| *alt* | Attributo | Descrizione alternativa alla visualizzazione. |
| *width* | Attributo | Larghezza. |
| *height* | Attributo | Altezza. |
| lyimage | #PCDATA | Codice LilyPond letterale. |
| *alt* | Attributo | Descrizione alternativa alla visualizzazione. |
| *width* | Attributo | Larghezza. |
| *height* | Attributo | Altezza. |
| teximage | #PCDATA | Codice TeX letterale. |
| *alt* | Attributo | Descrizione alternativa alla visualizzazione. |
| *width* | Attributo | Larghezza. |
| *height* | Attributo | Altezza. |
| lateximage | #PCDATA | Codice LaTeX letterale. |
| *alt* | Attributo | Descrizione alternativa alla visualizzazione. |
| *width* | Attributo | Larghezza. |
| *height* | Attributo | Altezza. |
| img | Vuoto | Immagine inserita in una riga. |
| *alt* | Attributo | Descrizione alternativa alla visualizzazione. |
| *imgfile* | Attributo | File contenente l'immagine, senza estensione. |
| *height* | Attributo | Altezza dell'immagine. |
| *width* | Attributo | Larghezza dell'immagine. |
| embimg | #PCDATA | Immagine incorporata; algoritmo Base64. |
| *alt* | Attributo | Descrizione alternativa alla visualizzazione. |
| *height* | Attributo | Altezza dell'immagine. |
| *width* | Attributo | Larghezza dell'immagine. |
| epsimg | #PCDATA | Codice EPS letterale. |
| *alt* | Attributo | Descrizione alternativa alla visualizzazione. |
| *width* | Attributo | Larghezza. |
| *height* | Attributo | Altezza. |

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| figimg | #PCDATA | Codice XFig letterale. |
| *alt* | Attributo | Descrizione alternativa alla visualizzazione. |
| *width* | Attributo | Larghezza. |
| *height* | Attributo | Altezza. |
| lyimg | #PCDATA | Codice LilyPond letterale. |
| *alt* | Attributo | Descrizione alternativa alla visualizzazione. |
| *width* | Attributo | Larghezza. |
| *height* | Attributo | Altezza. |
| teximg | #PCDATA | Codice TeX letterale. |
| *alt* | Attributo | Descrizione alternativa alla visualizzazione. |
| *width* | Attributo | Larghezza. |
| *height* | Attributo | Altezza. |
| latexing | #PCDATA | Codice LaTeX letterale. |
| *alt* | Attributo | Descrizione alternativa alla visualizzazione. |
| *width* | Attributo | Larghezza. |
| *height* | Attributo | Altezza. |

I nomi dei file indicati nell'attributo '**`imgfile`**' devono essere privi di estensione, intendendo implicitamente che questa sia '`.png`'.

Quando si inserisce il file di un'immagine, l'elemento relativo consente l'utilizzo degli attributi '**`height`**' e '**`width`**'. Evidentemente il primo permette di specificare l'altezza della figura e il secondo riguarda la larghezza. In linea di principio, i file di immagini hanno delle dimensioni, anche se queste sono espresse in pixel, ovvero in punti grafici. In generale conviene specificare l'altezza, oppure la larghezza, tenendo in considerazione il risultato per la composizione stampata, sapendo che l'informazione mancante viene determinata in modo relativo. Evidentemente, fissando entrambe le dimensioni, si ottiene un adattamento dell'immagine che non è necessariamente relativo.

Le dimensioni, ovvero le stringhe che si assegnano agli attributi citati, hanno una forma prestabilita:

***n unità_di_misura***

La composizione in HTML implica l'adattamento delle figure, in modo tale che la dimensione in punti grafici corrisponda al 200 % dei punti tipografici.[4] In pratica, nell'ambito di questa conversione, un punto grafico equivale a circa 0,0278 pollici, ovvero a 0,7055 mm. Questo tipo di rapporto è quello che ha dimostrato produrre la composizione HTML più vicina al risultato stampato.

Gli elementi per l'inserimento di immagini nel testo, come si vede dalla tabella 10.12, hanno un attributo denominato '**`alt`**'. Si tratta di un'informazione facoltativa, con la quale si descrive brevemente l'immagine. Questa informazione serve nella composizione HTML, per mostrare una descrizione minima in caso di problemi nella visualizzazione dell'immagine.

Oltre a immagini esterne, è possibile incorporare nel sorgente SGML diversi tipi di immagini: file trasformati secondo l'algoritmo Base64; codice EPS; codice XFig; codice TeX; codice LaTeX.

[4] Un punto tipografico, viene inteso qui come corrispondente a 1/72 di pollice, secondo la convenzione del linguaggio PostScript.

Per incorporare un'immagine codificata con l'algoritmo Base64 si può usare il programma Uuencode, oppure Mpack. Supponendo di utilizzare Uuencode e di volere inserire l'immagine contenuta nel file '`prova.jpg`', basta procedere come segue:

```
$ uuencode -m prova.jpg ciao > prova.uuencode
```

Quello che si ottiene in questo caso è il file '`prova.uuencode`', che può apparire simile al testo seguente, che è stato ridotto per comodità:

```
begin-base64 664 ciao
JSFQUy1BZG9iZS0yLjAKJSVDcmVhdG9yOiAiYmFyY29kZSIsIGxpYmJhcmNv
ZGUgc2FtcGxlIGZyb250ZW5kCiUgJSVEb2N1bWVudFBhcGVyU2l6ZXM6IGE0
...
...
b3cKMTA0LjAwIDEwLjAwIG1vdmV0byAoOSkgc2hvdwoKJSBFbmQgYmFyY29k
ZSBmb3IgIjk5MTIzNDU2Nzg5MCIKCiUlRW5kUGFnZQoKc2hvd3BhZ2UKJSVU
cmFpbGVyCiUlRU9GCgo=
====
```

Da questo file, ottenuto con Uuencode, va tolta la prima e l'ultima riga; il resto si può inserire in un elemento '**`embimg`**', oppure '**`embimage`**'. Vengono mostrati entrambi i casi.

```
<p>Bla bla bla
<embimg alt="Esempio" height="4mm">
<![CDATA[
JSFQUy1BZG9iZS0yLjAKJSVDcmVhdG9yOiAiYmFyY29kZSIsIGxpYmJhcmNv
ZGUgc2FtcGxlIGZyb250ZW5kCiUgJSVEb2N1bWVudFBhcGVyU2l6ZXM6IGE0
...
...
b3cKMTA0LjAwIDEwLjAwIG1vdmV0byAoOSkgc2hvdwoKJSBFbmQgYmFyY29k
ZSBmb3IgIjk5MTIzNDU2Nzg5MCIKCiUlRW5kUGFnZQoKc2hvd3BhZ2UKJSVU
cmFpbGVyCiUlRU9GCgo=
]]>
</embimg> bla bla bla.</p>
```

```
<figure id="f-esempio-1">
<fcaption>
    Figura <figureref>. Ecco il mio primo esempio.
</fcaption>
<embimage alt="Esempio" height="4cm">
<![CDATA[
JSFQUy1BZG9iZS0yLjAKJSVDcmVhdG9yOiAiYmFyY29kZSIsIGxpYmJhcmNv
ZGUgc2FtcGxlIGZyb250ZW5kCiUgJSVEb2N1bWVudFBhcGVyU2l6ZXM6IGE0
...
...
b3cKMTA0LjAwIDEwLjAwIG1vdmV0byAoOSkgc2hvdwoKJSBFbmQgYmFyY29k
ZSBmb3IgIjk5MTIzNDU2Nzg5MCIKCiUlRW5kUGFnZQoKc2hvd3BhZ2UKJSVU
cmFpbGVyCiUlRU9GCgo=
]]>
</embimage>
</figure>
```

Dal momento che si vuole evitare qualunque interpretazione SGML, può essere conveniente racchiudere il contenuto di questi elementi in una sezione marcata di tipo CDATA, così come si può vedere.

In modo analogo funzionano gli elementi '**`epsimg`**' e '**`epsimage`**', per quanto riguarda il codice EPS inserito direttamente nel sorgente. Vengono mostrati due esempi.

```
<p>Bla bla bla
<epsimg alt="Esempio" height="4mm">
<![CDATA[
%!PS-Adobe-2.0 EPSF-1.2
%%Creator: Pinco Pallino
%%BoundingBox: 0 0 500 500
%%Title: Un bel disegno
%%EndComments
...
...
showpage
%%Trailer
%%EOF
]]>
</epsimg> bla bla bla.</p>
```

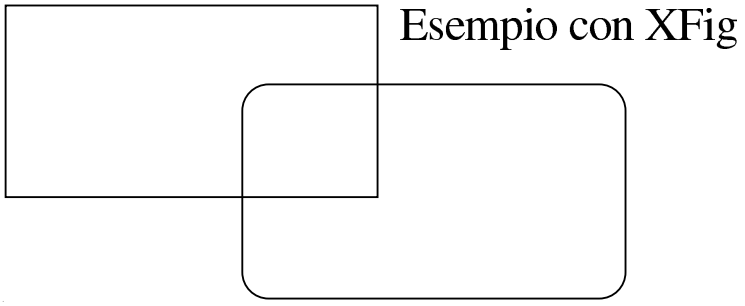
```
<figure id="f-esempio-1">
<fcaption>
    Figura <figureref>. Ecco il mio primo esempio.
</fcaption>
<epsimage alt="Esempio" height="4cm">
<![CDATA[
%!PS-Adobe-2.0 EPSF-1.2
%%Creator: Pinco Pallino
%%BoundingBox: 0 0 500 500
%%Title: Un bel disegno
%%EndComments
...
...
showpage
%%Trailer
%%EOF
]]>
</epsimage>
</figure>
```

Nello stesso modo, sono disponibili gli elementi '**figimg**' e '**figimage**', per quanto riguarda il codice XFig. Vengono mostrati due esempi e il risultato del secondo nella figura 10.1.

```
<p>Bla bla bla
<figimg alt="Esempio" height="4mm">
<![CDATA[
#FIG 3.2
Landscape
Center
Metric
A4
100.00
Single
-2
1200 2
2 2 0 1 0 7 50 0 -1 0.000 0 0 -1 0 0 5
         270 225 1755 225 1755 990 270 990 270 225
2 4 0 1 0 17 50 0 -1 0.000 0 0 7 0 0 5
         2745 1395 2745 540 1215 540 1215 1395 2745 1395
4 0 0 50 0 0 12 0.0000 4 180 1350 1845 360 Esempio con XFig\001
]]>
</figimg> bla bla bla.</p>
```

```
<figure id="f-esempio-1">
<fcaption>
    Figura <figureref>. Esempio con <special special="name">XFig</special>.
</fcaption>
<figimage alt="Esempio" height="4cm">
<![CDATA[
#FIG 3.2
Landscape
Center
Metric
A4
100.00
Single
-2
1200 2
2 2 0 1 0 7 50 0 -1 0.000 0 0 -1 0 0 5
         270 225 1755 225 1755 990 270 990 270 225
2 4 0 1 0 17 50 0 -1 0.000 0 0 7 0 0 5
         2745 1395 2745 540 1215 540 1215 1395 2745 1395
4 0 0 50 0 0 12 0.0000 4 180 1350 1845 360 Esempio con XFig\001
]]>
</figimage>
</figure>
```

Figura 10.1. Esempio con XFig.

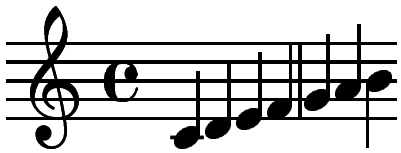



Sono disponibili gli elementi ‘**lyimg**’ e ‘**lyimage**’ per incorporare codice LilyPond. Vengono mostrati due esempi e il risultato del secondo nella figura 10.2.

```
<p>Bla bla bla
<lyimg alt="Esempio" height="5mm">
<![CDATA[
\version "1.6.6"
\header {
    tagline = ""
}
\score {
  \notes {c’ d’ e’ f’ g’ a’ b’}
  \paper {
    linewidth = 50
    pagenumber = "no"
  }
  \midi {}
}
]]>
</lyimg> bla bla bla.</p>
```

```
<figure id="f-esempio-1">
<fcaption>
    Figura <figureref>. Esempio con <special special="name">LilyPond</special>.
</fcaption>
<lyimage alt="Esempio" height="2cm">
<![CDATA[
\version "1.6.6"
\header {
    tagline = ""
}
\score {
  \notes {c' d' e' f' g' a' b'}
  \paper {
    linewidth = 50
    pagenumber = "no"
  }
  \midi {}
}
]]>
</lyimage>
</figure>
```

Figura 10.2. Esempio con LilyPond.



Si osservi che nella composizione in formato HTML, in corrispondenza dell'immagine che riproduce il codice musicale di LilyPond, se previsto, si raggiunge il file MIDI corrispondente come riferimento ipertestuale.

Infine, funzionano così anche gli elementi '**teximg**', '**teximage**', '**lateximg**' e '**lateximage**' per quanto riguarda il codice TeX e LaTeX inserito direttamente nel sorgente. Per la precisione, nel caso di '**teximg**' e '**teximage**', vengono aggiunte automaticamente all'inizio due istruzioni, '**\nonstopmode**' e '**\nopagenumbers**', inoltre, alla fine viene aggiunta l'istruzione '**\bye**'; invece, nel caso di '**lateximg**' e '**lateximage**' viene aggiunta l'istruzione '**\nonstopmode**' all'inizio e '**\end{document}**' alla fine.

Il codice LaTeX che viene inserito deve includere tutto il necessario a funzionare correttamente, ma l'aggiunta dell'istruzione '**\end{document}**' in modo automatico non può far male se questa è già stata inserita correttamente.

Segue un esempio riferito all'inclusione di codice TeX:

```
<figure id="f-esempio-1">
<fcaption>
    Figura <figureref>. Ecco una bella formula.
</fcaption>
<teximage alt="Esempio">
<![CDATA[
$$ \chi^2 = \sum_{i=1}^N
          \left (y_i - (a + b x_i)
          \over \sigma_i\right)^2 $$
]]>
</teximage>
</figure>
```

Figura 10.3. Ecco una bella formula.

$$\chi^2 = \sum_{i=1}^{N} \left( \frac{y_i - (a + bx_i)}{\sigma_i} \right)^2$$

Segue lo stesso esempio, realizzato con l'inclusione di codice LaTeX; si osservi in particolare la necessità di definire il tipo di documento e il tipo di pagina più semplice:

```
<figure id="f-esempio-2">
<fcaption>
    Figura <figureref>. Ecco un'altra bella formula.
</fcaption>
<lateximage alt="Esempio">
<![CDATA[
\documentclass{article}
\pagestyle{empty}
\begin{document}
$$ \chi^2 = \sum_{i=1}^N
          \left (y_i - (a + b x_i)
          \over \sigma_i\right)^2 $$
\end{document}
]]>
</lateximage>
</figure>
```

Figura 10.4. Ecco un'altra bella formula.

$$\chi^2 = \sum_{i=1}^{N} \left( \frac{y_i - (a + bx_i)}{\sigma_i} \right)^2$$

### 10.5.6 Tabelle

Come nel caso delle figure, le tabelle sono organizzate in modo da poter essere rappresentate da qualunque cosa: una tabella come si è abituati di solito, oppure dei blocchi di testo, anche preformattato, come '**`pre`**' e '**`verbatimpre`**'.

L'involucro di una tabella funziona in modo simile a quello di una figura:

```
<table id="t-esempio-1">
<tcaption>
    Tabella <tableref>. Ecco il mio primo esempio.
</tcaption>
...
...
</table>
```

Anche l'elemento '**table**' possiede gli attributi '**id**' e '**pos**', con lo stesso significato che hanno nell'elemento '**figure**'. Nello stesso modo funziona la didascalia, che in questo caso è delimitata dall'elemento '**tcaption**', mentre il riferimento all'ultima tabella avviene con l'elemento '**tableref**'.

A parte la possibilità di disegnare la tabella usando blocchi di testo normali, la tabella tipica incorpora l'elemento '**tabular**':

```
<table id="t-esempio-1">
<tcaption>
    Tabella <tableref>. Ecco il mio primo esempio.
</tcaption>
<tabular col="2">
<thead>
<trow>  Dispositivo     <colsep> Descrizione                    </trow>
</thead>
<tbody>
<trow> /dev/fd0         <colsep> Prima unità a dischetti.       </trow>
<trow> /dev/hda         <colsep> Primo disco fisso ATA.         </trow>
<trow> /dev/hdb         <colsep> Secondo disco fisso ATA.       </trow>
<trow> /dev/sda         <colsep> Primo disco SCSI.              </trow>
<trow> /dev/lp0         <colsep> Prima porta parallela.         </trow>
<trow> /dev/ttyS0       <colsep> Prima porta seriale.           </trow>
</tbody>
</tabular>
</table>
```

L'esempio mostrato è sufficientemente completo: l'elemento '**tabular**' ha un attributo obbligatorio, '**col**', con il quale è necessario dichiarare subito la quantità di colonne che compone la tabella. Le righe della tabella sono raggruppate in due gruppi: l'intestazione, delimitata dall'elemento '**thead**', e il corpo, delimitato dall'elemento '**tbody**'. Le righe sono definite dall'elemento '**trow**' e la separazione tra una colonna e l'altra avviene con l'elemento vuoto '**colsep**'.

Tabella 10.13. Elementi SGML che servono a rappresentare le tabelle.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| table | | Involucro di una tabella. |
| *id* | Attributo | Ancora di riferimento per la tabella. |
| *pos* | Attributo | '**fixed**', '**float**'. |
| *split* | Attributo | Non si può dividere = '**0**' (predefinito); si può dividere automaticamente = '**1**'. |
| tcaption | %inline; | Didascalia. |
| tabular | | Descrizione del reticolo di righe e colonne. |
| *col* | Attributo | Quantità di colonne presenti. |
| *columnfractions* | Attributo | Frazioni orizzontali per le colonne. |
| *printedfontsize* | Attributo | Dimensione del carattere da usare. |
| *border* | Attributo | Normale = '**0**' (predefinito); caselle bordate = '**1**'. |
| thead | trow | Righe di intestazione. |

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| trow | | Riga. |
| colsep | | Separazione tra le colonne. |
| tbody | trow | Righe del corpo. |
| trow | | Riga. |
| colsep | | Separazione tra le colonne. |

La gestione delle tabelle di Alml è un po' limitata; in situazioni eccezionali, si può valutare anche la possibilità di realizzare tabelle HTML utilizzando l'elemento '**`html`**', come si vede nell'esempio di tabella 10.25. Tuttavia, si deve ricordare che si tratta di codice esterno, per cui non si possono inserire elementi tipici di Alml, ma solo codice HTML; inoltre, la trasformazione in forma di testo puro di una tabella HTML complessa non avviene sempre nel modo corretto.

L'esempio seguente mostra il caso di una tabella in cui le celle possono contenere più di una riga. Si vede il risultato in 10.14.

```
<table id="t-tex-controllo-paragrafo-comune">
<tcaption>

    Tabella <tableref>. Esempio di tabella un po' più complessa.

</tcaption>
<tabular col="3" columnfractions="0.2 0.4 0.4" border="1">
<thead>
  <trow>Parola di controllo
<colsep>Competenza
<colsep>Condizione o valore predefinito
</trow>
</thead>
<tbody>
  <trow>\hoffset
<colsep>Posizione iniziale dei paragrafi nella pagina.
<colsep><num>0</num>
</trow>
  <trow>\hsize
<colsep>Larghezza del paragrafo a partire da <samp>\hoffset</samp>.
<colsep><num>6,5</num> pollici
</trow>
  <trow>\parindent
<colsep>Rientro della prima riga.
<colsep><num>20</num> punti
</trow>
  <trow>\baselineskip
<colsep>Distanza tra la base di una riga e la base della riga successiva.
<colsep><num>12</num> punti
</trow>
  <trow>\parskip
<colsep>Distanza aggiuntiva tra i paragrafi.
<colsep><num>0</num>
</trow>
  <trow>\raggedright
<colsep>Allinea il testo a sinistra.
<colsep>allineato simultaneamente a sinistra e a destra
</trow>
  <trow>\leftskip
<colsep>Rientro sinistro complessivo.
<colsep><num>0</num>
</trow>
  <trow>\rightskip
```

```
<colsep>Rientro destro complessivo.
<colsep><num>0</num>
</trow>
</tbody>
</tabular>
</table>
```

Tabella 10.14. Esempio di tabella un po' più complessa.

| Parola di controllo | Competenza | Condizione o valore predefinito |
|---|---|---|
| \hoffset | Posizione iniziale dei paragrafi nella pagina. | 0 |
| \hsize | Larghezza del paragrafo a partire da '**\hoffset**'. | 6,5 pollici |
| \parindent | Rientro della prima riga. | 20 punti |
| \baselineskip | Distanza tra la base di una riga e la base della riga successiva. | 12 punti |
| \parskip | Distanza aggiuntiva tra i paragrafi. | 0 |
| \raggedright | Allinea il testo a sinistra. | allineato simultaneamente a sinistra e a destra |
| \leftskip | Rientro sinistro complessivo. | 0 |
| \rightskip | Rientro destro complessivo. | 0 |

Eccezionalmente (purché si utilizzi l'attributo '**columnfractions**'), è possibile inserire nelle celle alcuni elementi che rappresentano blocchi di testo; per esempio: '**syntax**', '**command**', '**pre**' e '**verbatimpre**'. Ciò dovrebbe consentire l'uso delle tabelle per realizzare degli schemi riassuntivi riferiti a comandi sintassi o simili. Si osservi l'esempio seguente, rappresentato successivamente dalla tabella 10.15.[5]

```
<table id="alml-esempio-sintassi-in-tabella">
<tabular col="2" columnfractions="0.618 0.382" border="1">
<thead>
  <trow>Comando
<colsep>Descrizione
</trow>
</thead>
<tbody>
  <trow><syntax>mbadblock <var>unità_dos</var></syntax>
<colsep>Scandisce un'unità &DOS; alla ricerca di settori difettosi.
</trow>
  <trow><syntax>mcd <synsqb><var>directory_dos</var></synsqb></syntax>
<colsep>Permette di modificare o conoscere la directory corrente delle unità &DOS;.
</trow>
  <trow><syntax>mdel <var>file_dos</var><synellipsis></syntax>
<colsep>Cancella i file &DOS; indicati come argomento.
</trow>
  <trow><syntax>mdeltree <var>directory_dos</var><synellipsis></syntax>
<colsep>Cancella le directory &DOS; indicate come argomento.
</trow>
  <trow><syntax>mmd <var>directory_dos</var><synellipsis></syntax>
<colsep>Crea le directory &DOS; indicate come argomento.
</trow>
  <trow><syntax>mmove <var>origine_dos</var><synellipsis> <var>destinazione_dos</var></syntax>
<colsep>Sposta o rinomina uno o più file e directory.
</trow>
  <trow><syntax>mrd <var>directory_dos</var><synellipsis></syntax>
<colsep>Elimina le directory indicate come argomento, purché siano vuote.
</trow>
  <trow><syntax>mren <var>origine_dos</var><synellipsis> <var>destinazione_dos</var></syntax>
<colsep>Rinomina o sposta uno o più file e directory.
</trow>
</tbody>
```

[5] La scelta del rapporto tra le due colonne della tabella, 61,8 % e 38,2 %, rappresenta quello che è noto come «rapporto aureo».

```
</tabular>
</table>
```

| Comando | Descrizione |
|---|---|
| `mbadblock` ***unità_dos*** | Scandisce un'unità Dos alla ricerca di settori difettosi. |
| `mcd` [***directory_dos***] | Permette di modificare o conoscere la directory corrente delle unità Dos. |
| `mdel` ***file_dos***... | Cancella i file Dos indicati come argomento. |
| `mdeltree` ***directory_dos***... | Cancella le directory Dos indicate come argomento. |
| `mmd` ***directory_dos***... | Crea le directory Dos indicate come argomento. |
| `mmove` ***origine_dos***... ***destinazione_dos*** | Sposta o rinomina uno o più file e directory. |
| `mrd` ***directory_dos***... | Elimina le directory indicate come argomento, purché siano vuote. |
| `mren` ***origine_dos***... ***destinazione_dos*** | Rinomina o sposta uno o più file e directory. |

Le tabella molto lunghe possono essere realizzate in modo da consentire il salto pagina, utilizzando l'attributo '**`split`**'. In ogni caso, perché ci possa essere una tabella suddivisibile tra le pagine, è necessario che questa non sia fluttuante.

### 10.5.7 Listati

Come le figure e le tabelle, i listati possono essere rappresentati da qualunque cosa; di solito si tratta di blocchi di testo preformattato, come '**`pre`**' e '**`verbatimpre`**'. L'involucro di un listato funziona in modo simile a quello di una figura:

```
<listing id="l-esempio-1">
<lcaption>
    Listato <listingref>. Ecco il mio primo esempio.
</lcaption>
...
...
</listing>
```

Anche l'elemento '**`listing`**' possiede gli attributi '**`id`**', '**`pos`**' e '**`sep`**', con lo stesso significato che hanno nell'elemento '**`figure`**'. Nello stesso modo funziona la didascalia, che in questo caso è delimitata dall'elemento '**`lcaption`**', mentre il riferimento all'ultimo listato avviene con l'elemento '**`listingref`**'.

Tabella 10.16. Elementi SGML che servono a rappresentare i listati.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| listing | | Involucro di un listato. |
| *id* | Attributo | Ancora di riferimento per il listato. |
| *pos* | Attributo | '**`fixed`**', '**`float`**'. |
| *sep* | Attributo | '**`none`**', '**`rule`**', '**`border`**'. |
| *split* | Attributo | Non si può dividere = '**`0`**' (predefinito); si può dividere automaticamente = '**`1`**'. |
| lcaption | %inline; | Didascalia. |

### 10.5.8 Riferimenti incrociati e ipertestuali

I riferimenti incrociati si realizzano attraverso l'indicazione di ancore (o etichette se si preferisce il termine) e di puntatori a tali ancore. Esistono diversi modi per definire un'ancora e un riferimento a questa: tutti gli elementi che dispongono di un attributo '**`id`**', sono ancore oppure sono puntatori alle ancore.

Fino a questo punto sono stati descritti gli elementi che delimitano i titoli dei volumi, delle parti, dei capitoli e delle sezioni; inoltre sono stati visti gli elementi che avvolgono le figure e le tabelle. Tutti questi sono ancore a cui si può puntare, ma per inserire un'ancora nel testo normale, è possibile usare l'elemento vuoto '**`anchor`**', anche questo provvisto di attributo '**`id`**'.

Esistono quattro elementi vuoti per fare riferimento alle ancore: '**`sectionref`**', per ottenere un riferimento alla sezione in cui si trova l'ancora; '**`figureref`**' per fare riferimento a una figura; '**`tableref`**' per fare riferimento a una tabella; '**`listingref`**' per fare riferimento a un listato.

In particolare, gli elementi '**`figureref`**', '**`tableref`**' e '**`listingref`**' possono essere usati anche senza l'attributo '**`id`**' per fare riferimento all'ultima ancora di una figura, di una tabella o di un listato, come è già stato mostrato nell'uso delle didascalie.

Quando si realizza un documento che può includere o meno una certa porzione a cui puntano alcuni riferimenti, per evitare che vengano mostrati questi collegamenti mancanti, si può usare l'elemento '**`ifref`**', con il quale si delimita la parte da non comporre se manca il riferimento indicato nell'attributo '**`id`**'.

Tabella 10.17. Gestione dei riferimenti incrociati.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| tomeheading | | Titolo di un volume. |
| *id* | Attributo | Ancora di riferimento per il titolo del volume. |
| h0 | | Titolo di una parte. |
| *id* | Attributo | Ancora di riferimento per il titolo della parte. |
| h1 | | Titolo di un capitolo. |
| *id* | Attributo | Ancora di riferimento per il titolo di un capitolo. |
| h2 | | Titolo di una sezione. |
| *id* | Attributo | Ancora di riferimento per il titolo di una sezione. |
| h3 | | Titolo di una sottosezione. |
| *id* | Attributo | Ancora di riferimento per il titolo di una sottosezione. |
| h4 | | Titolo di una sotto-sottosezione. |
| *id* | Attributo | Ancora per il titolo di una sotto-sottosezione. |
| anchor | Vuoto | Ancora inserita nel testo. |
| *id* | Attributo | Stringa di identificazione dell'ancora. |
| sectionref | Vuoto | Riferimento a un'ancora del testo. |
| *id* | Attributo | Stringa a cui si fa riferimento. |
| figure | | Involucro di una figura. |
| *id* | Attributo | Ancora di riferimento per la figura. |
| figureref | Vuoto | Riferimento a un'ancora di una figura. |
| *id* | Attributo | Stringa a cui si fa riferimento. |
| table | | Involucro di una tabella. |
| *id* | Attributo | Ancora di riferimento per la tabella. |

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| tableref | Vuoto | Riferimento a un'ancora di una tabella. |
| *id* | Attributo | Stringa a cui si fa riferimento. |
| listing | | Involucro di un listato. |
| *id* | Attributo | Ancora di riferimento per il listato. |
| listingref | Vuoto | Riferimento a un'ancora di un listato. |
| *id* | Attributo | Stringa a cui si fa riferimento. |
| ifref | %inline; o %block; | Delimita un'area da comporre solo se l'ancora esiste veramente. |
| *id* | Attributo | Stringa a cui si fa riferimento. |

### 10.5.9 Note e piè pagina

Alml prevede l'utilizzo di tre tipi di annotazioni: avvertimenti che devono risaltare in un riquadro e due tipi di note a piè pagina. Le note evidenziate sono indicate all'interno di un elemento '**`frame`**', mentre quelle a piè pagina sono inserite nell'elemento '**`footnote`**', oppure '**`blockfootnote`**'.

Le note a piè pagina normali sono quelle dell'elemento '**`footnote`**', che si colloca all'interno delle righe; al contrario, '**`blockfootnote`**' rappresenta un blocco di testo, che rimane solo per compatibilità con il passato.

```
<frame>
    <p>Attenzione! Si tratta di un'operazione rischiosa.</p>
</frame>
```

L'esempio precedente mostra l'utilizzo di un riquadro, mentre quello successivo mostra l'uso di un piè pagina normale.

```
<p>Bla bla bla<footnote>Questa parola si ripete.</footnote> bla bla...</p>
```

Tabella 10.18. Annotazioni a vario titolo.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| frame | %block; | Riquadro. |
| blockfootnote | %inline; | Piè pagina tra i blocchi di testo. |
| footnote | %inline; | Piè pagina all'interno di una riga di testo. |

### 10.5.10 Riferimenti esterni e citazioni

Alcuni elementi sono specializzati per fare riferimento a qualcosa di esterno. Il caso più comune riguarda l'elemento '**`uri`**', con il quale si indica un URI:

```
<p>Bla bla bla <uri><![CDATA[http://www.brot.dg]]></uri> bla bla...</p>
```

Per indicare il riferimento a una pagina di manuale, si può usare l'elemento '**`man`**', in modo da ottenere una rappresentazione uguale a quella tradizionale:

```
<p>Bla bla bla <man>ls<mansect>1</mansect></man> bla bla...</p>
```

La tabella 10.19 riepiloga questi e altri elementi affini.

Tabella 10.19. Riferimenti esterni.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| uri | CDATA | Riferimento a un URI esterno. |
| uristr | #PCDATA | Riferimento a un URI che non funziona. |
| blockquote | %block;, quoteinfo | Citazione. |
| quoteinfo | %inline; | Informazioni sulla citazione. |
| bibref | %inline; | Titolo di un documento. |
| man | #PCDATA, mansect | Pagina di manuale. |
| mansect | #PCDATA | Numero della sezione. |

L'elemento '**uristr**' è una variante di '**uri**', con lo scopo di non generare un riferimento ipertestuale. Ciò può servire per rappresentare un indirizzo di fantasia, oppure un indirizzo reale che non è più valido. Si possono indicare in questo modo anche i nomi di dominio.

L'elemento '**blockquote**' è previsto per delimitare una citazione in uno o più blocchi. Alla fine dell'elemento '**blockquote**' è prevista la possibilità di usare un solo elemento '**quoteinfo**', con lo scopo di contenere informazioni relative alla citazione:

```
<blockquote>
    %block;
    …
    [<quoteinfo>%inline;…</quoteinfo>]
</blockquote>
```

## 10.5.11 Altre inserzioni particolari

Sono disponibili altri elementi di importanza minore. Si tratta di '**br**', '**hr**', '**newpage**', '**bottompage**', '**heightrequired**' e '**navlink**'. I primi due emulano gli elementi corrispondenti dell'HTML, interrompendo una riga e inserendo una linea orizzontale rispettivamente.

L'elemento '**newpage**' richiede un salto pagina, se il tipo di composizione lo consente.

L'elemento '**bottompage**' serve per definire un gruppo di blocchi di testo da rappresentare nella parte bassa della pagina, nella composizione per la stampa. In pratica, si usa '**bottompage**' per delimitare informazioni legali nella seconda pagina relativa dei volumi:

```
<tomeheading>Bla bla bla</tomeheading>

<bottompage>
    <p>Copyright &copy; Pinco Pallino...</p>

    <p>Bla bla bla...</p>
</bottompage>
```

L'elemento '**heightrequired**' serve nella composizione per la stampa, a garantire che sia disponibile una certa quantità di spazio (un'altezza minima prima della fine della pagina), in mancanza del quale viene inserito un salto pagina.

Tabella 10.20. Inserzioni varie.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| br | Vuoto | Interruzione della riga. |
| hr | Vuoto | Riga orizzontale di separazione. |
| newpage | Vuoto | Salto pagina se ammissibile. |
| bottompage | %block; | Testo da rappresentare nella parte bassa della pagina. |
| heightrequired | Vuoto | Serve a richiedere espressamente la presenza di una certa quantità di spazio prima della fine della pagina. |
| *height* | Attributo | Altezza minima richiesta prima della fine della pagina. |
| navlink | #PCDATA | Riferimento ipertestuale per la navigazione HTML. |

L'elemento '**navlink**' consente di aggiungere nella composizione HTML un riferimento ipertestuale fisso, in tutte le pagine, allo scopo di raggiungere facilmente la posizione in cui l'elemento stesso viene inserito. Si osservi l'esempio seguente:

```
<h1>
Indice analitico
</h1>

<navlink>indice analitico</navlink>

<printindex index="main">

</index>
```

Si tratta dell'inserimento dell'indice analitico, con l'aggiunta di un riferimento ipertestuale fisso nelle pagine della composizione HTML.

## 10.6 Tracciamento di informazioni particolari

Diversi tipi di elementi nella struttura di Alml sono predisposti per accumulare informazioni da restituire a richiesta. La situazione più semplice è data dalla gestione degli indici analitici, dove con l'elemento '**indexentry**' si inserisce una voce nell'indice analitico generale o in un altro individuato da un nome libero:

```
<h1>
I colori dell'arcobaleno
<indexentry>arcobaleno</indexentry>
<indexentry><code>color</code></indexentry>
</h1>
```

L'elemento '**indexentry**' appartiene al gruppo di quelli che possono essere inseriti all'interno di una riga; nell'esempio si vede la situazione tipica in cui lo si inserisce nel testo di un titolo. In questo caso, sono state indicate due voci dell'indice analitico generale: la parola «arcobaleno» viene inserita in modo normale, mentre la parola «color» viene inserita con un carattere dattilografico.

Ogni indice analitico ha un nome e quello generale, o predefinito, corrisponde a '**main**'. L'esempio mostrato sopra sarebbe perfettamente equivalente a quello seguente:

```
<h1>
I colori dell'arcobaleno
<indexentry index="main">arcobaleno</indexentry>
<indexentry index="main"><code>color</code></indexentry>
</h1>
```

Per recuperare l'elenco di un indice analitico si utilizza l'elemento '**printindex**', in cui, lo stesso attributo '**index**' permette di stabilire quale indice estrapolare.

Tabella 10.21. Gestione degli indici analitici.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| indexentry | #PCDATA \| code | Dichiarazione di una voce per l'indice analitico. |
| *index* | Attributo | Nome dell'indice analitico in cui inserire la voce. |
| special | #PCDATA | Termine speciale. |
| *special* | Attributo | Nome dell'indice analitico in cui inserire la voce. |
| printindex | Vuoto | Inserisce l'elenco dell'indice analitico richiesto. |
| *index* | Attributo | Nome dell'indice analitico richiesto. |
| *indexcontext* | Attributo | Specifica un contesto tra: '**all**', '**tome**', '**part**', '**chapter**'. È predefinito il contesto '**all**', che richiede l'indice completo. |
| *indexref* | Attributo | Specifica in che modo devono apparire i riferimenti: '**default**', '**section**'. In pratica, con la parola chiave '**section**' si impone di mostrare numeri di sezione e non le pagine. |

Esiste anche un altro elemento che inserisce voci negli indici analitici; si tratta di '**special**', che inserisce una voce nell'indice corrispondente al nome indicato con l'attributo che ha lo stesso nome: '**special**'.

### 10.6.1 Caratteristiche del software e di altri «lavori»

La struttura di Alml dispone di un elemento speciale che si può inserire nel testo lineare, il cui scopo è quello di annotare alcune informazioni sul software e su lavori simili. Si osservi l'esempio seguente:

```
<p>Stiamo parlando di Mpage,
<workinfo>
<workname>Mpage</workname>
<worklicense>licenza speciale che non ammette le modifiche</worklicense>
<worklicensetext>

    <p>Permission is granted to anyone to make or distribute verbatim
    copies of this document as received, in any medium, provided that
    this copyright notice is preserved, and that the distributor grants
    the recipient permission for further redistribution as permitted by
    this notice.</p>

</worklicensetext>
</workinfo>
un programma che si occupa di...</p>
```

Solo gli elementi '**workname**' e '**worklicense**' sono obbligatori, dal momento che il loro contenuto appare in un piè pagina locale. L'elemento '**worklicensetext**' è facoltativo e può essere utile per annotare una licenza unica, per la quale non possa essere individuato un riferimento standard; inoltre, un altro elemento, '**worknotes**', permette di annotare qualcosa al riguardo.

Dove lo si ritiene più opportuno, si può collocare l'elemento '**printworkinfo**', per ottenere l'elenco ordinato di queste informazioni accumulate.

Tabella 10.22. Tracciamento di informazioni sul software citato.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| workinfo | | Dichiarazione del blocco di informazioni. |
| workname | #PCDATA | Nome del software o di altro lavoro. |
| worklicense | #PCDATA | Denominazione o descrizione breve della licenza. |
| worklicensetext | %block; | Testo della licenza specifica. |
| worknotes | %block; | Annotazioni. |
| printworkinfo | Vuoto | Inserisce le informazioni accumulate in modo ordinato. |
| *workinforef* | Attributo | Specifica in che modo devono apparire i riferimenti: '**default**', '**section**'. In pratica, con la parola chiave '**section**' si impone di mostrare numeri di sezione e non le pagine. |

## 10.6.2 Informazioni su sezioni specifiche del documento

In situazioni particolari, potrebbe essere necessario, o anche solo utile, tenere traccia dell'origine di una sezione del documento, assieme a delle annotazioni a vario titolo. Per questo si può utilizzare l'elemento '**docinfo**', che questa volta costituisce un blocco.

```
<docinfo>

    <dl>
    <dt>2002.09.15</dt>
    <dd>

        <p>Il testo viene aggiornato nel contenuto, con l'inserimento
        della sezione «bla bla bla», da parte di Caio Cai
        (caio@brot.dg).</p>

    </dd>
    <dt>2002.09.08</dt>
    <dd>

        <p>Il testo viene modificato per adeguarlo alla nuova veste
        grafica dell'opera, per opera di Caio Cai (caio@brot.dg);
        il contenuto rimane invariato.</p>

    </dd>
    <dt>2002.02.02</dt>
    <dd>

        <p>Il testo originale è di Tizio Tizi e risale al 2002.02.02. Nello
        stesso giorno, il testo ha subito qualche aggiustamento per opera di
        Caio Cai (caio@brot.dg), con il consenso dell'autore.</p>

    </dd>
```

```
    </dl>

</docinfo>
```

L'esempio mostra in particolare l'uso dell'elemento '**docinfo**' per annotare lo storico delle modifiche fatte su quella porzione di documento; come si può vedere, vengono indicate prima le azioni più recenti, ma questo dipende solo da una scelta organizzativa.

Per ottenere l'elenco delle informazioni accumulate in questo modo, si utilizza l'elemento vuoto '**printdocinfo**'.

Tabella 10.23. Tracciamento di informazioni su sezioni particolari del documento globale.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| docinfo | | Annotazioni sul documento (volume, parte, capitolo o sezione inferiore). |
| printdocinfo | Vuoto | Inserisce nel testo le informazioni accumulate. |

## 10.6.3 Condizioni particolari per il contenuto di una sezione

È previsto un contenitore speciale per indicare le condizioni particolari che riguardano una certa sezione (anche un volume intero). Si tratta dell'elemento '**specialcondition**', all'interno del quale può eventualmente apparire l'elemento vuoto '**nomod**':

```
<specialcondition><nomod>non è consentita la modifica di questa
sezione</specialcondition>
```

L'esempio dovrebbe rendere l'idea della cosa. Il testo contenuto nell'elemento '**specialcondition**' viene mostrato effettivamente, utilizzando un carattere un po' diverso da quello normale, in modo da risaltare.

L'elemento vuoto '**nomod**' serve per tenere traccia in particolare di quelle sezioni che non possono essere modificate. Evidentemente, può essere utile solo se il documento, nella sua globalità, è inteso come modificabile, in base alle condizioni della licenza. In generale non dovrebbe essere necessario, perché se nella sezione che non si può modificare è scritto chiaramente come stanno le cose al riguardo, non dovrebbe servire alcun elenco di tali sezioni; tuttavia, in questo modo, è possibile poi ottenere un elenco dettagliato di tutte le sezioni che non possono essere modificate, con l'elemento vuoto '**printnomod**'.

Tabella 10.24. Annotazione delle condizioni particolari di una sezione.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| specialcondition | #PCDATA \| nomod | Dichiarazione di condizioni particolari. |
| nomod | Vuoto | Annotazione di sezione non modificabile. |
| printnomod | Vuoto | Elenco delle sezioni non modificabili. |

## 10.7 Inserimento letterale di codice TeX e HTML, con eventuale inserimento condizionato

In situazioni eccezionali, può essere conveniente l'inserimento di codice scritto secondo il linguaggio di composizione che si trova al di sotto della struttura SGML di Alml. Lo scopo di Alml non è quello di mantenere un legame sicuro con TeX e HTML, tuttavia viene lasciata aperta questa possibilità.

Si pensi all'eventuale necessità di inserire qualcosa di particolare nella composizione HTML, per esempio per mettere un contatore di accesso, o altri tipi di inserzioni ritenute utili per qualche ragione.

Per risolvere questo problema si possono usare due elementi speciali: '**tex**' e '**html**'. Come si può intuire, il primo elemento è fatto per racchiudere codice TeX o LaTeX; il secondo serve per includere codice HTML.

Dal momento che si vuole evitare qualunque interpretazione SGML, conviene racchiudere il contenuto di questi elementi in una sezione marcata di tipo CDATA. Si osservi l'esempio seguente riferito a codice HTML:

```
<html><![CDATA[
    <hr>
    <p><a href="http://www.digits.com/">Web-Counter: </a><a
    href="http://www.digits.com/"><img
    src="http://counter.digits.com/wc/tizio345"
    alt="counter"></a></p>
]]></html>
```

A fianco di questo problema, sta poi la possibilità di delimitare facilmente dei blocchi di sorgente che debbano essere presi in considerazione solo se la composizione avviene attraverso una trasformazione in TeX o in HTML. In pratica, si utilizzano rispettivamente gli elementi '**iftex**' e '**ifhtml**'. Questi elementi non sono indispensabili, perché l'SGML offre già un meccanismo di controllo dell'elaborazione del sorgente, attraverso le sezioni marcate; tuttavia, servono per completare e concludere il problema degli elementi contenenti codice speciale TeX o HTML.

Il codice HTML può essere rappresentato in parte anche quando la composizione avviene attraverso TeX, per mezzo di HTML2ps. In pratica, con il codice HTML si ottiene un'immagine che viene poi incorporata nel sorgente TeX. Questa estensione serve specialmente per consentire la realizzazione di tabelle più complesse di quanto consenta Alml con il suo elemento '**tabular**'. Si osservi l'esempio seguente, che viene poi rappresentato nella tabella 10.25:

```
<table id="t-alml-incorporazione-tabella-html">
<tcaption>
    Tabella <tableref>. Incorporazione di codice HTML per rappresentare
    una tabella complessa.
</tcaption>
<html width=15cm>
<![CDATA[
<table border="1">
  <thead>
    <tr>
      <td rowspan="2"><p>Denominazione della porta seriale su i386 nei sistemi
        Dos</p>
      </td>
      <td colspan="2"><p>Risorse</p>
      </td>
      <td rowspan="2"><p>File di dispositivo nei sistemi GNU/Linux</p>
      </td>
      <td rowspan="2"><p>Annotazioni</p>
      </td>
    </tr>
    <tr>
      <td><p>IRQ</p>
      </td>
      <td><p>I/O</p>
      </td>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td><p>COM1:</p>
      </td>
      <td rowspan="2"><p align="center">4</p>
      </td>
      <td><p>3F8<sub>16</sub></p>
      </td>
      <td><p>/dev/ttyS0</p>
      </td>
      <td rowspan="2"><p>La prima e la terza porta seriale condividono lo
        stesso IRQ.</p>
      </td>
    </tr>
    <tr>
      <td><p>COM3:</p>
      </td>
      <td><p>3E8<sub>16</sub></p>
      </td>
      <td><p>/dev/ttyS2</p>
      </td>
    </tr>
    <tr>
      <td><p>COM2:</p>
      </td>
      <td rowspan="2"><p align="center">3</p>
      </td>
      <td><p align="right">2F8<sub>16</sub></p>
      </td>
      <td><p>/dev/ttyS0</p>
      </td>
      <td rowspan="2"><p>La seconda e la quarta porta seriale condividono lo
        stesso IRQ.</p>
      </td>
    </tr>
    <tr>
      <td><p>COM4:</p>
```

```
      </td>
      <td><p align="right">2E8<sub>16</sub></p>
      </td>
      <td><p>/dev/ttyS2</p>
      </td>
    </tr>
  </tbody>
</table>
]]>
</html>
</table>
```

Tabella 10.25. Incorporazione di codice HTML per rappresentare una tabella complessa.

| Denominazione della porta seriale su i386 nei sistemi Dos | Risorse | | File di dispositivo nei sistemi GNU/Linux | Annotazioni |
|---|---|---|---|---|
| | IRQ | I/O | | |
| COM1: | 4 | $3F8_{16}$ | /dev/ttyS0 | La prima e la terza porta seriale condividono lo stesso IRQ. |
| COM3: | | $3E8_{16}$ | /dev/ttyS2 | |
| COM2: | 3 | $2F8_{16}$ | /dev/ttyS0 | La seconda e la quarta porta seriale condividono lo stesso IRQ. |
| COM4: | | $2E8_{16}$ | /dev/ttyS2 | |

Si osservi nell'esempio l'uso dell'attributo '**`width`**'. Precisamente, l'elemento '**`html`**' consente l'uso degli attributi '**`width`**' e '**`height`**' per stabilire le dimensioni dell'oggetto HTML importato nella composizione stampata. In questo caso, è stata specificata la larghezza, corrispondente allo spazio orizzontale a disposizione, in modo che l'altezza venga adattata automaticamente, mantenendo lo stesso rapporto.

Tabella 10.26. Inserimento letterale di codice TeX e HTML; inserimento condizionato in base al tipo di composizione.

| Elemento o attributo | Contenuto | Descrizione |
|---|---|---|
| html | #PCDATA | Codice HTML letterale. |
| *width* | Attributo | Larghezza nella composizione stampata. |
| *height* | Attributo | Altezza nella composizione stampata. |
| tex | #PCDATA | Codice TeX o LaTeX letterale. |
| ifhtml | %block; | Blocco condizionato alla composizione in HTML. |
| iftex | %block; | Blocco condizionato alla composizione in TeX. |

Si rammenti che mentre quanto contenuto nell'elemento '**`html`**' appare sia nella composizione per la stampa, sia nella composizione HTML, l'elemento '**`tex`**' genera un risultato utile solo nella composizione per la stampa.
Per quanto riguarda il caso particolare dell'elemento '**`tex`**', si tenga in considerazione piuttosto la possibilità di usare gli elementi '**`teximg`**' e '**`teximage`**', che generano un risultato visibile anche nel formato HTML finale, attraverso la trasformazione automatica in forma di immagine.

## 10.8 Definizione alternativa della suddivisione del documento

Alml è pensato per la realizzazione di documenti di grandi dimensioni. In questo senso, la sua struttura normale è quella di un libro, articolato in capitoli che si possono raggruppare in parti e volumi. Queste suddivisioni prevedono una denominazione attribuita automaticamente, corrispondente a «capitolo», «parte» e «tomo»; eventualmente, se questa struttura va definita invece attraverso termini differenti, si possono sostituire queste parole con altre più appropriate.

Per questo si usano gli elementi '**chapterdefinition**', '**partdefinition**' e '**tomedefinition**', all'interno delle informazioni amministrative. L'esempio seguente dovrebbe permettere di comprendere il problema; per la precisione si tratta di una rivista telematica ipotetica:

```
<head>
    <admin>
        <description>Rivista di informatica libera</description>
        <keywords>informatica libera, software libero</keywords>
        <chapterdefinition>articolo</chapterdefinition>
        <partdefinition>numero</partdefinition>
        <tomedefinition>anno</tomedefinition>
    </admin>
    <title>RIL, rivista di informatica libera</title>
    <author>Pinco Pallino &lt;pinco.pallino@brot.dg&gt;</author>
    <date>2011.11.11</date>
    <legal>
        <p>Copyright &copy; Pinco Pallino, &lt;pinco.pallino@brot.dg&gt;</p>
    </legal>
    <maincontents levels="2">Table of contents</maincontents>
</head>
```

Si può osservare che le parole «articolo», «numero» e «anno», sono state inserite usando lettere minuscole e in forma singolare. Ciò è necessario, perché l'iniziale maiuscola viene ottenuta automaticamente quando opportuno; inoltre, questi termini vengono usati sempre quando si fa riferimento a un solo oggetto.

La numerazione dei volumi, delle parti e dei capitoli è indipendente, per cui non ci si può aspettare che al cambio di un volume o di una parte, i capitoli riprendano la numerazione a partire da uno.

## 10.9 Riferimento alla larghezza del testo

In generale, non esiste la possibilità di attribuire agli attributi '**width**' di immagini e simili, un riferimento alla larghezza effettivamente a disposizione. In altri termini, non esiste la possibilità di fare riferimento a ciò che per LaTeX può essere l'istruzione '**\textwidth**'. In questo modo, se si decide in un momento successivo di modificare lo spazio orizzontale utilizzabile all'interno delle pagine, si può essere costretti a verificare tutte le dimensioni di questi oggetti particolari. Per risolvere l'inconveniente si possono definire delle entità interne all'inizio del documento, come nell'esempio seguente:

```
<!DOCTYPE ALML PUBLIC "-//D.G.//DTD Alml//EN"
[
<!ENTITY BODYWIDTH     "15cm">
<!ENTITY BODYWIDTH0.75 "11.25cm">
<!ENTITY BODYWIDTH0.50 "7.5cm">
 ...
]>
```

Successivamente si potrebbe usare la prima di queste entità per dichiarare la larghezza del corpo della pagina:

```
<head>
    <admin>
        ...
        <printedpagesize type="bodywidth">&BODYWIDTH;</printedpagesize>
        ...
    </admin>
```

Quindi, nello stesso modo per le immagini e altri oggetti che si traducono nell'incorporazione di immagini che devono avere una larghezza uguale o proporzionale all'ampiezza massima disponibile:

```
<figure id="f-esempio-1">
<fcaption>
    Figura <figureref>. Esempio con <special special="name">XFig</special>.
</fcaption>
<figimage alt="Esempio" height="&BODYWIDTH;">
<![CDATA[
#FIG 3.2
Landscape
Center
Metric
A4
100.00
Single
-2
1200 2
2 2 0 1 0 7 50 0 -1 0.000 0 0 -1 0 0 5
         270 225 1755 225 1755 990 270 990 270 225
2 4 0 1 0 17 50 0 -1 0.000 0 0 7 0 0 5
         2745 1395 2745 540 1215 540 1215 1395 2745 1395
4 0 0 50 0 0 12 0.0000 4 180 1350 1845 360 Esempio con XFig\001
]]>
</figimage>
</figure>
<p>Bla bla bla bla...</p>
<figure id="f-esempio-2">
<fcaption>
    Figura <figureref>. Un altro esempio più piccolo.
</fcaption>
<figimage alt="Esempio piccolo" height="&BODYWIDTH0.75;">
<![CDATA[
#FIG 3.2
Landscape
Center
Metric
A4
100.00
Single
-2
1200 2
2 2 0 1 0 7 50 0 -1 0.000 0 0 -1 0 0 5
         270 225 1755 225 1755 990 270 990 270 225
2 4 0 1 0 17 50 0 -1 0.000 0 0 7 0 0 5
         2745 1395 2745 540 1215 540 1215 1395 2745 1395
4 0 0 50 0 0 12 0.0000 4 180 1350 1845 360 Esempio con XFig\001
]]>
</figimage>
</figure>
```

# Entità ISO gestite da Alml

Nel seguito vengono mostrate alcune tabelle che riportano lo stato attuale del supporto dato da Alml alle entità ISO standard. Ciò che non è disponibile, appare come racchiuso tra parentesi quadre.

Tabella 11.1. Entità ISOnum: *numeric and special graphic*. Prima parte.

| SGML macro | Risultato | Descrizione in inglese |
|---|---|---|
| ½ | ½ | fraction one-half |
| ½ | ½ | fraction one-half |
| ¼ | ¼ | fraction one-quarter |
| ¾ | ¾ | fraction three-quarters |
| ⅛ | [frac18] | fraction one-eighth |
| ⅜ | [frac38] | fraction three-eighths |
| ⅝ | [frac58] | fraction five-eighths |
| ⅞ | [frac78] | fraction seven-eighths |
| ¹ | ¹ | superscript one |
| ² | ² | superscript two |
| ³ | ³ | superscript three |
| + | + | plus sign |
| ± | ± | plus-or-minus sign |
| < | < | less-than sign |
| = | = | equals sign |
| > | > | greater-than sign |
| ÷ | ÷ | divide sign |
| × | × | multiply sign |
| ¤ | ¤ | general currency sign |
| £ | £ | pound sign |
| $ | $ | dollar sign |
| ¢ | ¢ | cent sign |
| ¥ | ¥ | yen sign |
| # | # | number sign |
| % | % | percent sign |
| & | & | ampersand |
| * | * | asterisk |
| @ | @ | commercial at |
| [ | [ | left square bracket |
| \ | \ | reverse solidus |
| ] | ] | right square bracket |
| { | { | left curly bracket |
| ― | [horbar] | horizontal bar |
| | | \| | vertical bar |
| } | } | right curly bracket |

Tabella 11.2. Entità ISOnum: *numeric and special graphic*. Seconda parte.

| SGML macro | Risultato | Descrizione in inglese |
|---|---|---|
| µ | $\mu$ | micro sign |
| Ω | [ohm ] | ohm sign |
| ° | ° | degree sign |
| º | º | ordinal indicator, masculine |
| ª | ª | ordinal indicator, feminine |
| § | § | section sign |
| ¶ | ¶ | pilcrow (paragraph sign) |
| · | · | middle dot |

| SGML macro | Risultato | Descrizione in inglese |
|---|---|---|
| ← | [larr ] | leftward arrow |
| → | [rarr ] | rightward arrow |
| ↑ | [uarr ] | upward arrow |
| ↓ | [darr ] | downward arrow |
| © | © | copyright sign |
| ® | ® | registered sign |
| ™ | [trade ] | trade mark sign |
| ¦ | ¦ | broken (vertical) bar |
| ¬ | ¬ | not sign |
| ♪ | [sung ] | music note (sung text sign) |
| ! | ! | exclamation mark |
| ¡ | ¡ | inverted exclamation mark |
| " | " | quotation mark |
| ' | ’ | apostrophe |
| ( | ( | left parenthesis |
| ) | ) | right parenthesis |
| , | , | comma |
| _ | _ | low line |
| ‐ | - | hyphen |
| . | . | full stop, period |
| / | / | solidus |
| : | : | colon |
| ; | ; | semicolon |
| ? | ? | question mark |
| ¿ | ¿ | inverted question mark |
| « | « | angle quotation mark, left |
| » | » | angle quotation mark, right |
| ‘ | ‘ | single quotation mark, left |
| ’ | ’ | single quotation mark, right |
| “ | [ldquo ] | double quotation mark, left |
| ” | [rdquo ] | double quotation mark, right |
|   | | no break (required) space |
|  | | soft hyphen |

Tabella 11.3. Entità ISOtech: *general technical*. Prima parte.

| SGML macro | Risultato | Descrizione in inglese |
|---|---|---|
| ℵ | [aleph ] | aleph, Hebrew |
| ∧ | [and ] | logical and |
| &ang90; | [ang90 ] | right (90 degree) angle |
| ∢ | [angsph] | angle-spherical |
| ≈ | [ap ] | approximate |
| ∵ | [becaus] | because |
| ⊥ | [bottom] | perpendicular |
| ∩ | [cap ] | intersection |
| ≅ | [cong ] | congruent with |
| ∮ | [conint] | contour integral operator |
| ∪ | [cup ] | union or logical sum |
| ≡ | [equiv ] | identical with |
| ∃ | [exist ] | at least one exists |
| ∀ | [forall] | for all |
| ƒ | [fnof ] | function of (italic small f) |
| ≥ | ≥ | greater-than-or-equal |
| ⇔ | [iff ] | if and only if |
| ∞ | [infin ] | infinity |
| ∫ | [int ] | integral operator |

| SGML macro | Risultato | Descrizione in inglese |
|---|---|---|
| ∈ | [isin ] | set membership |
| ⟨ | [lang ] | left angle bracket |
| ⇐ | [lArr ] | is implied by |
| ≤ | $\leq$ | less-than-or-equal |
| − | - | minus sign |
| ∓ | [mnplus] | minus-or-plus sign |
| ∇ | [nabla ] | del, Hamilton operator |
| ≠ | [ne ] | not equal |
| ∋ | [ni ] | contains |
| ∨ | [or ] | logical or |

Tabella 11.4. Entità ISOtech: *general technical*. Seconda parte.

| SGML macro | Risultato | Descrizione in inglese |
|---|---|---|
| ∥ | [par ] | parallel |
| ∂ | [part ] | partial differential |
| ‰ | [permil] | per thousand |
| ⊥ | [perp ] | perpendicular |
| ′ | [prime ] | prime or minute |
| ″ | [Prime ] | double prime or second |
| ∝ | [prop ] | is proportional to |
| √ | [radic ] | radical |
| ⟩ | [rang ] | right angle bracket |
| ⇒ | [rArr ] | implies |
| ∼ | [sim ] | similar |
| ≃ | [sime ] | similar, equals |
| □ | [square] | square |
| ⊂ | [sub ] | subset or is implied by |
| ⊆ | [sube ] | subset, equals |
| ⊃ | [sup ] | superset or implies |
| ⊇ | [supe ] | superset, equals |
| ∴ | [there4] | therefore |
| ‖ | [Verbar] | dbl vertical bar |
| Å | [angst ] | capital A, ring |
| ℬ | [bernou] | bernoulli function (script capital B) |
| ∘ | [compfn] | composite function (small circle) |
| ¨ | [Dot ] | dieresis or umlaut mark |
| ⃜ | [DotDot] | four dots above |
| ℋ | [hamilt] | hamiltonian (script capital H) |
| ℒ | [lagran] | lagrangian (script capital L) |
| ∗ | [lowast] | low asterisk |
| ∉ | [notin ] | negated set membership |
| ℴ | [order ] | order of (script small o) |
| ℳ | [phmmat] | physics M-matrix (script capital M) |
| ⃛ | [tdot ] | three dots above |
| ‴ | [tprime] | triple prime |
| ≙ | [wedgeq] | corresponds to (wedge, equals) |

Tabella 11.5. Entità ISOlat1: *added latin 1*. Prima parte.

| SGML macro | Risultato | Descrizione in inglese |
|---|---|---|
| á | á | small a, acute accent |
| Á | Á | capital A, acute accent |
| â | â | small a, circumflex accent |
| Â | Â | capital A, circumflex accent |
| à | à | small a, grave accent |
| À | À | capital A, grave accent |
| å | å | small a, ring |
| Å | Å | capital A, ring |
| ã | ã | small a, tilde |
| Ã | Ã | capital A, tilde |
| ä | ä | small a, dieresis or umlaut mark |
| Ä | Ä | capital A, dieresis or umlaut mark |
| æ | æ | small ae diphthong (ligature) |
| Æ | Æ | capital AE diphthong (ligature) |
| ç | ç | small c, cedilla |
| Ç | Ç | capital C, cedilla |
| ð | ð | small eth, Icelandic |
| Ð | Ð | capital Eth, Icelandic |
| é | é | small e, acute accent |
| É | É | capital E, acute accent |
| ê | ê | small e, circumflex accent |
| Ê | Ê | capital E, circumflex accent |
| è | è | small e, grave accent |
| È | È | capital E, grave accent |
| ë | ë | small e, dieresis or umlaut mark |
| Ë | Ë | capital E, dieresis or umlaut mark |
| í | í | small i, acute accent |
| Í | Í | capital I, acute accent |
| î | î | small i, circumflex accent |
| Î | Î | capital I, circumflex accent |
| ì | ì | small i, grave accent |
| Ì | Ì | capital I, grave accent |
| ï | ï | small i, dieresis or umlaut mark |
| Ï | Ï | capital I, dieresis or umlaut mark |

Tabella 11.6. Entità ISOlat1: *added latin 1*. Seconda parte.

| SGML macro | Risultato | Descrizione in inglese |
|---|---|---|
| ñ | ñ | small n, tilde |
| Ñ | Ñ | capital N, tilde |
| ó | ó | small o, acute accent |
| Ó | Ó | capital O, acute accent |
| ô | ô | small o, circumflex accent |
| Ô | Ô | capital O, circumflex accent |
| ò | ò | small o, grave accent |
| Ò | Ò | capital O, grave accent |
| ø | ø | small o, slash |
| Ø | Ø | capital O, slash |
| õ | õ | small o, tilde |
| Õ | Õ | capital O, tilde |
| ö | ö | small o, dieresis or umlaut mark |
| Ö | Ö | capital O, dieresis or umlaut mark |
| ß | ß | small sharp s, German (sz ligature) |
| þ | þ | small thorn, Icelandic |

| SGML macro | Risultato | Descrizione in inglese |
|---|---|---|
| Þ | Þ | capital THORN, Icelandic |
| ú | ú | small u, acute accent |
| Ú | Ú | capital U, acute accent |
| û | û | small u, circumflex accent |
| Û | Û | capital U, circumflex accent |
| ù | ù | small u, grave accent |
| Ù | Ù | capital U, grave accent |
| ü | ü | small u, dieresis or umlaut mark |
| Ü | Ü | capital U, dieresis or umlaut mark |
| ý | ý | small y, acute accent |
| Ý | Ý | capital Y, acute accent |
| ÿ | ÿ | small y, dieresis or umlaut mark |

# Stile di scrittura del sorgente

Il DTD di Alml suggerisce una logica nella stesura del sorgente. In questo capitolo si annotano dei suggerimenti sulla sistemazione degli elementi nel sorgente, allo scopo di ottenere una struttura ordinata, in funzione delle caratteristiche di questi.

## 12.1 Blocchi di testo e rientri

In generale, un blocco di testo viene scritto a partire dalla prima colonna del file, oppure viene incolonnato più a destra, di quattro caratteri alla volta, se si tratta di un sottoblocco di qualche tipo. Si osservi l'esempio seguente:

```
<frame>

    <p>Bisogna fare attenzione alle...
    ...
    ...</p>

</frame>
```

L'elemento '**`frame`**' serve a contenere uno o più blocchi interni; questi vanno indicati con un rientro.

> Alla regola del rientro devono fare eccezione quei blocchi in cui lo spazio iniziale ha significato. In questo modo, gli elementi '**`pre`**', '**`verbatimpre`**', '**`asciiart`**' e '**`syntax`**' devono iniziare sempre dalla prima colonna.

I blocchi di testo con un contenuto di tipo '**`%inline;`**', ovvero testo lineare ed elementi interni a questo, dovrebbero mostrare la loro natura, avvolgendo il testo stesso, senza aggiungere rientri ulteriori. Per esempio, si usa l'elemento '**`p`**' in questo modo:

```
<p>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla...</p>
```

Al contrario, sarebbe spiacevole scrivere una cosa del genere:

```
<p>
    Bla bla bla bla...
</p>
```

I blocchi di testo, allineati in base alla necessità, vanno poi organizzati in modo da evitare di uscire dalla portata visiva di uno schermo normale; in pratica dovrebbero trovarsi entro le prime 80 colonne, come nell'esempio seguente:

```
<frame>

    <p>I blocchi di testo, allineati in base alla necessità, vanno poi
    organizzati in modo da evitare di uscire dalla portata visiva di uno
    schermo normale; in pratica dovrebbero trovarsi entro le prime
    <num>80</num> colonne.</p>

</frame>
```

Per favorire l'uso di funzionalità adatte del proprio programma di scrittura, allo scopo di reimpaginare i paragrafi e gli altri blocchi di testo, è necessario staccare i blocchi di testo tra di loro e dal loro contenitore.

### 12.1.1 Elenchi

Gli elenchi di Alml sono definiti in modo da contenere sempre blocchi di testo. In tal modo, la struttura più coerente con quanto affermato a proposito dei rientri e dell'impaginazione dei blocchi, è quella dello schema seguente per ciò che riguarda gli elenchi puntati e numerati:

```
<ul>|<ol>
<li>

    blocco

    [blocco]
    …

</li>
[<li>

    blocco

    [blocco]
    …

</li>]
…
</ul>|</ol>
```

Per gli elenchi descrittivi, la situazione è abbastanza simile:

```
<dl>
<dt>voce</dt>
<dd>

    blocco

    [blocco]
    …

</dd>
[<dt>voce</dt>
<dd>

    blocco

    [blocco]
    …

</dd>]
…
</dl>
```

In modo equivalente si comporta anche l'elemento '**segment**':

```
<segment>
<segmenthead>titolo</segmenthead>

    blocco

    [blocco]
    ...

[<segmenthead>titolo</segmenthead>

    blocco

    [blocco]
    ...]

</segment>
```

## 12.2 Figure e tabelle

Le figure interne al testo seguono la sorte di tutti gli altri elementi del genere, mentre le figure contenute nell'elemento '**figure**' possono spostarsi sulla superficie della pagina. In questo senso, conviene indicarle sempre a partire dalla prima colonna, anche quando si chiede espressamente che rimangano fisse nella posizione in cui si trovano nel sorgente. L'elemento '**figure**' è predisposto per contenere altri elementi, che però non è il caso di indicare con rientri. L'esempio seguente mostra la situazione comune in cui la figura è rappresentata dall'elemento '**image**'; in particolare merita attenzione la didascalia.

```
<figure id="f-esempio-1">
<fcaption>

    Figura <figureref>. Bla bla bla...

</fcaption>
<image imgfile="esempio-1" height="4cm">
</figure>
```

In effetti, la didascalia è contenuta in un elemento '**fcaption**' che costituisce un blocco di testo. In precedenza è stata descritta la regola per cui i blocchi di testo devono essere realizzati ponendo il marcatore iniziale e quello finale in aderenza al testo contenuto, reimpaginando il tutto in base all'incolonnamento. Tuttavia, quello che si vede nell'esempio è lo stile proposto, che vale quindi come eccezione nel caso delle didascalie di figure, tabelle e listati.

Per le tabelle valgono le stesse considerazioni in relazione alle didascalie, mentre si propone una struttura particolare per l'elenco degli elementi che compongono le varie righe.

```
<table id="t-alml-isolat1-2">
<tcaption>

    Tabella <tableref>. Entità <special special="name">ISOlat1</special>:
    <bibref>added latin 1</bibref>. Seconda parte.

</tcaption>
<tabular col="3">
<thead>
  <trow>SGML macro
<colsep>Risultato
<colsep>Descrizione in inglese
</trow>
</thead>
<tbody>
```

```
  <trow>&amp;ntilde;
<colsep>&ntilde;
<colsep>small n, tilde
</trow>
...
  <trow>&amp;yuml;
<colsep>&yuml;
<colsep>small y, dieresis or umlaut mark
</trow>
</tbody>
</tabular>
</table>
```

L'esempio mostra una situazione tipica. Si può osservare l'allineamento particolare del marcatore '**<trow>**' per avere il testo di tutte le celle della tabella allineato sulla stessa colonna del sorgente.

Allo scopo di facilitare la riorganizzazione di una tabella, è bene evitare di spezzare le righe di testo di una cella, quando queste superano la dimensione dello schermo.

## 12.3 Titoli

Gli elementi che contengono il titolo di una sezione (come per esempio '**tomeheading**', '**h0**', '**h1**', '**h2**', ecc.), vengono indicati nel sorgente secondo la struttura seguente, che mostra in particolare il caso del capitolo:

```
<h1 [id="stringa_identificativa"]>
titolo
[<indexentry [index="indice"]>voce_indice</indexentry>]
...
</h1>
```

Per facilitare un rielaborazione eventuale del sorgente, dovuta a una modifica del DTD di Alml, conviene lasciare il testo del titolo su una sola riga, anche se questo può essere lungo; inoltre, per lo stesso motivo, anche se il contenuto dell'elemento del titolo è di tipo lineare, conviene separare i marcatori dal testo del titolo, così come si vede dallo schema mostrato. Infine, per facilitare l'organizzazione delle voci da inserire nell'indice analitico, conviene collocare gli elementi '**indexentry**' preferibilmente nell'elemento del titolo, dopo il testo che lo descrive, in modo da guidare il lettore all'inizio della sezione che contiene la parola cercata.

## 12.4 Sezioni marcate

Le sezioni marcate devono essere delimitate correttamente e quando queste sono annidate, si possono creare problemi nel riconoscere la fine di questa o quella sezione. Per evitare ambiguità, è bene segnalare la macro dell'entità parametrica relativa:

```
<[%nome_entità_parametrica; [

blocco_protetto

[blocco_protetto]
...

]]><!--%nome_entità_parametrica; -->
```

Quando una sezione marcata controlla una porzione di testo normale, è sufficiente che sia evidente l'ambito della sezione stessa. Per esempio:

```
 <p>Bla bla bla <![%SENZACONTROLLO;[ciao ciao]]> bla bla bla...</p>
```

# Gestione di «Appunti di informatica libera»

Questo capitolo descrive l'organizzazione del sorgente di *Appunti di informatica libera*, in modo da consentire una comprensione migliore del funzionamento di Alml.

## 13.1 Articolazione dei file del sorgente

Il sorgente di *Appunti di informatica libera* è composto da un file principale, molto grande, che fa riferimento ad altri file esterni per vari motivi:

```
.
|-- antologie/
|   `-- ...
|
|-- figure/
|   `-- *.png
|
|-- ortografia/
|   |-- erroriecczioni
|   |-- minimo.aff
|   |-- minimo.hash
|   |-- minimo.sml
|   |-- particolari
|   `-- vocabolario
|
|-- ospiti/
|    `-- lavoro_ospitato/
|         `...
|
|-- .textchk.rules      --> ortografia/erroriecczioni
|-- .textchk.special    --> ortografia/particolari
|-- Makefile
`-- a2-nnnnn.sgml
```

I file '`.textchk.rules`' e '`.textchk.special`', ovvero '`ortografia/erroriecczioni`' e '`ortografia/particolari`', servono per l'uso di Textchk; mentre i file rimanenti nella directory '`ortografia/`' riguardano Ispell.

## 13.2 Inclusione selettiva dei file esterni ed entità speciali

L'inclusione dei file esterni, nel blocco principale, avviene per mezzo di istruzioni SGML del tipo seguente, dove si dichiara un'entità a cui si abbina il contenuto di un file intero:

```
<!ENTITY CONTR-LDR-free       SYSTEM "ospiti/LDR/inclusi/LDR-corpo-free.sgml">
```

Altri pezzi ricorrenti di codice SGML sono dichiarati come entità interne, come questa:

```
>!ENTITY ALCOPY.TEXT
'

    <endofchapter>&ALOPERA; 2003.06.29 >![%SHORTTYPESETTING;[test]]> ---
    <em>Copyright &copy; &ALPERIODO; Daniele Giacomini --
    daniele @ swlibero.org</em><![%ANNOTAZIONI;[ --- si prega di non diffondere
    questa bozza]]></endofchapter>

'>
```

A seconda della circostanza, può essere necessario includere tali file o tali entità, oppure evitare la cosa. Per esempio, in una composizione che genera un file HTML unico non è il caso di ripetere certe informazioni sul copyright alla fine di ogni capitolo. Per questa e per altre ragioni, si utilizzano delle entità parametriche che nel sorgente vengono dichiarate in modo da disabilitarle:

```
<!ENTITY % HTML            "IGNORE">
<!ENTITY % PLAINHTML       "IGNORE">
<!ENTITY % POSTSCRIPT      "IGNORE">
<!ENTITY % PLAINPOSTSCRIPT "IGNORE">
<!ENTITY % LEGGIMI         "IGNORE">
<!ENTITY % ANNOTAZIONI     "IGNORE">
<!ENTITY % SENZACONTROLLO  "IGNORE">
<!ENTITY % OBSOLETO        "IGNORE">
```

Queste entità parametriche controllano la dichiarazione di entità normali e l'inclusione di testo normale, come si può vedere nell'estratto semplificato che segue:

```
<![%POSTSCRIPT;[
    <!ENTITY ALCOPYINGTOMO   "&ALCOPYINGTOMO.TEXT;">
    <!ENTITY ALCOPYINGPARTE  "&ALCOPYINGPARTE.TEXT;">
    <!ENTITY ALCOPY          "&ALCOPY.TEXT;">
    <!ENTITY ALDEDICA        "&ALDEDICA.TEXT;">
    <!ENTITY LDRCOPYINGTOMO  SYSTEM "ospiti/LDR/formalita/copying-tomo.sgml">
    <!ENTITY LDRCOPYINGPARTE SYSTEM "ospiti/LDR/formalita/copying-parte.sgml">
    <!ENTITY LDRCOPY         SYSTEM "ospiti/LDR/formalita/copy.sgml">
]]>

<![%PLAINPOSTSCRIPT;[
    <!ENTITY ALCOPYINGTOMO   "">
    <!ENTITY ALCOPYINGPARTE  "">
    <!ENTITY ALCOPY          "">
    <!ENTITY ALDEDICA        "&ALDEDICA.TEXT;">
    <!ENTITY LDRCOPYINGTOMO  SYSTEM "ospiti/LDR/formalita/copying-tomo.sgml">
    <!ENTITY LDRCOPYINGPARTE "">
    <!ENTITY LDRCOPY         "">
]]>
```

Se tutte le entità parametriche viste in precedenza restano al valore originale ('**IGNORE**'), nessuna delle dichiarazioni che si vedono qui viene presa in considerazione. Se invece una di queste entità contiene il valore '**INCLUDE**', allora le dichiarazioni relative hanno significato.

Il sistema controlla l'abilitazione di queste entità parametriche attraverso l'opzione '**--sgml-include=*entità_parametrica***', come per esempio nel comando necessario a generare una composizione in PostScript:

```
$ alml --ps --verbose ↩
↪--sgml-include=POSTSCRIPT ↩
↪--sgml-include=SENZACONTROLLO ↩
↪--sgml-include=OBSOLETO ↩
↪mio_file.sgml
```

Questa abilitazione preventiva prende il sopravvento sulla dichiarazione di esclusione (‘**IGNORE**’) interna al sorgente e si ottiene il risultato desiderato.

Anche la dichiarazione delle entità normali segue la regola per cui vale ciò che è stato definito per primo. Pertanto, per evitare problemi, dopo la dichiarazione condizionata all’attivazione delle entità parametriche, viene ripetuta una dichiarazione di tali entità in modo predefinito:

```
<!ENTITY ALCOPYINGTOMO          "&ALCOPYINGTOMO.TEXT;">
<!ENTITY ALCOPYINGPARTE         "&ALCOPYINGPARTE.TEXT;">
<!ENTITY ALCOPY                 "&ALCOPY.TEXT;">
<!ENTITY ALDEDICA               "&ALDEDICA.TEXT;">
<!ENTITY LDRCOPYINGTOMO  SYSTEM "ospiti/LDR/formalita/copying-tomo.sgml">
<!ENTITY LDRCOPYINGPARTE SYSTEM "ospiti/LDR/formalita/copying-parte.sgml">
<!ENTITY LDRCOPY         SYSTEM "ospiti/LDR/formalita/copy.sgml">
```

Successivamente, nel corpo del file principale appare il richiamo alle entità relative per indicare il punto di inserzione del loro contenuto:

```
<tomeheading>
Primo approccio, architettura e filosofia del sistema operativo
</tomeheading>
&ALCOPYINGTOMO;

<h0>
Il software e le licenze
</h0>
&ALCOPYINGPARTE;

<h1>
...
```

Le tabelle 13.1 e 13.2 riepilogano le entità parametriche che controllano il sorgente di *Appunti di informatica libera* e le entità normali più importanti.

Tabella 13.1. Significato delle entità parametriche più importanti, usate nel sorgente di *Appunti di informatica libera*.

| Macro SGML | Significato se attiva |
|---|---|
| `%HTML;` | Composizione HTML normale. |
| `%PLAINHTML;` | Composizione HTML su una pagina unica. |
| `%POSTSCRIPT;` | Composizione PostScript o PDF normale. |
| `%PLAINPOSTSCRIPT;` | Composizione PostScript speciale per risparmiare spazio. |
| `%LEGGIMI;` | Controlla l’inclusione di alcune note introduttive. |
| `%ANNOTAZIONI;` | Composizione con annotazioni per uso interno. |
| `%SENZACONTROLLO;` | Composizione completa di ciò che non viene controllato ortograficamente. |
| `%OBSOLETO;` | Composizione completa di informazioni ritenute obsolete o incomplete. |

Tabella 13.2. Significato di alcune entità importanti, usate nel sorgente di *Appunti di informatica libera*.

| Macro SGML | Contenuto |
|---|---|
| `&ALOPERA;` | Il nome dell’opera. |
| `&ALOPERAEMAIL;` | L’indirizzo o gli indirizzi di posta elettronica di riferimento. |
| `&ALPERIODO;` | L’anno o gli anni del copyright. |
| `&ALEDIZIONE;` | Edizione, scritta possibilmente come data. |

## 13.3 Composizione guidata con il file-make

Il pacchetto dei sorgenti di *Appunti di informatica libera* include il file '`Makefile`', per facilitare la composizione dell'opera. La tabella 13.3 riepiloga i comandi principali.

Tabella 13.3. Comandi relativi al file-make di *Appunti di informatica libera*.

| Comando | Risultato |
|---|---|
| make clean | Ripulisce da tutti i file non indispensabili. |
| make check | Analizza la sintassi SGML. |
| make spell | Utilizza Ispell per l'analisi del vocabolario. |
| make textchk | Utilizza Textchk per l'analisi sintattica. |
| make urichk | Utilizza Checkbot per il controllo degli URI. |
| make draftdvi | Composizione bozza in DVI. |
| make dvi | Composizione finale in DVI. |
| make draftps | Composizione bozza in PostScript. |
| make ps | Composizione finale in PostScript. |
| make longps | Composizione finale in PostScript compatto e ridotto. |
| make extralongps | Composizione finale in PostScript compatto e ultra ridotto. |
| make largeps | Composizione finale in PostScript, orizzontale, compatto e ridotto. |
| make extralargeps | Composizione finale in PostScript, orizzontale, compatto e ultra ridotto. |
| make draftpdf | Composizione bozza in PDF. |
| make pdf | Composizione finale in PDF. |
| make drafthtml | Composizione bozza in HTML. |
| make html | Composizione finale in HTML. |
| make html-text | Composizione finale in HTML a pagina singola. |
| make text | Composizione finale in formato testo puro. |

# Convenzioni di «Appunti di informatica libera»

Questo capitolo raccoglie alcune convenzioni importanti relative all'opera *Appunti di informatica libera*. Le annotazioni sulla terminologia sono separate in un altro capitolo.

## 14.1 Unità di misura e moltiplicatori

In informatica si utilizzano delle unità di misura e dei moltiplicatori ben conosciuti, ma senza uno standard simbolico ben definito. Nel testo di questo documento si usano le convenzioni elencate nel seguito.

In particolare è bene distinguere tra il nome di un'unità di misura e il simbolo che la rappresenta: quando si parla dell'unità si usa il nome esteso, minuscolo; quando si indica un valore si deve usare il simbolo. In altri termini, si può parlare di hertz in generale, ma poi si indicano ***n*** Hz per indicarne una quantità precisa.

Quando si nominano i prefissi moltiplicatori, come «mega», «giga» e «tera», si usano le iniziali minuscole anche se il simbolo corrispondente è dato dalla loro iniziale maiuscola.

| Unità di misura | Descrizione |
|---|---|
| byte, Kibyte, Mibyte, Gibyte,<br>bit, Kibit, Mibit, Gibit | L'unità byte viene indicata al minuscolo, di seguito al suo moltiplicatore eventuale. In particolare: «Ki» sta per $2^{10}$ = 1024; «Mi» sta per $2^{20}$ = 1048576; «Gi» sta per $2^{30}$ = 1073741824. L'unità di misura, con il suo moltiplicatore, viene indicata dopo e staccata dalla quantità a cui si riferisce. |
| bit/s, kbit/s, Mbit/s | L'unità bit/s (nota comunemente come bps, ovvero *Bit per second*) viene indicata al minuscolo, di seguito al suo moltiplicatore eventuale. In questo caso si utilizzano i moltiplicatori standard del SI: «k» sta per $10^3$ = 1000; «M» sta per $10^6$ = 1000000; «G» sta per $10^9$ = 1000000000. È importante ricordare che la lettera «k» deve essere minuscola.<br>In generale, è preferibile la notazione bit/s rispetto a bps, perché la seconda è in realtà un'abbreviazione e come tale sconsigliabile secondo il SI. A questo proposito, si può leggere *Guide for the Use of the International System of Units (SI)* edito dal NIST (*National institute of standards and technology*), <*http://physics.nist.gov/cuu/pdf/sp811.pdf*>, in particolare la sezione 6.1.8: *Unacceptability of abbreviations for units*. |
| Hz, kHz, MHz, GHz, THz | L'unità «hertz», il cui simbolo è «Hz», viene indicata nel modo che si vede, di seguito al suo moltiplicatore eventuale. In questo caso si utilizzano i moltiplicatori tradizionali: «k» sta per $10^3$ = 1000; «M» sta per $10^6$ = 1000000; «G» sta per $10^9$ = 1000000000; «T» sta per $10^{12}$ = 1000000000000. È importante ricordare che la lettera «k» deve essere minuscola.<br>Le unità di misura del SI, si nominano senza iniziale maiuscola. Tuttavia, il simbolo attribuito all'unità di misura è stato espresso con un'iniziale maiuscola quando questo derivava dal nome di una persona. Per esempio, questo è il caso di Hertz, di Alessandro Volta e di altri. |
| Ex | La grandezza Ex rappresenta l'altezza di una lettera «x», nell'ambito del sistema di composizione tipografica utilizzato. Viene indicata nel testo in questo modo, con l'iniziale maiuscola, per evitare confusione. Nel caso della misura relativa alla lettera «M» maiuscola, si usa il termine quadratone. |

## 14.2 Casi particolari di testo che non viene enfatizzato

Alle volte verrebbe da enfatizzare di tutto. Qui si annotano le cose che per regola non vengono enfatizzate.

- **Valori numerici**

  I valori numerici di qualunque sistema di numerazione non vengono enfatizzati e i valori espressi in base diversa da 10 si indicano come si vede qui: $11 = 0B_{16} = 13_8 = 1011_2$. In particolare, le lettere alfabetiche utilizzate per le basi di numerazione superiori a 10, sono maiuscole.

- **Classi di indirizzi IPv4**

  Le classi di indirizzi IPv4 sono definite da lettere alfabetiche maiuscole che qui non vengono enfatizzate.

- **Indirizzi IPv4**

  Gli indirizzi numerici IPv4, a ottetti, vengono rappresentati così come sono, senza enfatizzazioni, utilizzando eventualmente il simbolo '*' per rappresentare l'indifferenza del valore di uno o più ottetti.

- **Indirizzi IPv6**

  Gli indirizzi numerici IPv6 vengono rappresentati così come sono, senza enfatizzazioni, utilizzando lettere minuscole.

- **Denominazione dei record di risorsa nel DNS**

  Le sigle usate nel DNS per identificare i record di risorsa dei file di definizione delle zone, sono scritti usando lettere maiuscole, senza enfatizzazioni.

- **Comandi del modem**

  I comandi AT e gli altri comandi dei modem vengono indicati utilizzando lettere maiuscole e senza enfatizzazioni. Ci possono essere eccezioni a questa regola, per esempio quando il contesto fa riferimento a una stringa che in quel caso particolare corrisponde proprio a un comando da inviare al modem.

## 14.3 Valori numerici in lettera e in cifre

I valori numerici da zero a nove vengono rappresentati preferibilmente in lettere, soprattutto per evitare ambiguità nella lettura, a meno che si presentino le condizioni seguenti:

- il numero è seguito da un simbolo (secondo il SI o anche altre convenzioni), per cui si preferisce lasciarlo espresso in cifre;
- il numero fa parte di un intervallo, dove l'altro valore è composto da due o più cifre, così si lascia in cifra anche il primo, dal momento che non ci possono essere ambiguità.

## 14.4 Distinzione nell'uso dei nomi degli applicativi

In generale, in questo documento, i nomi riferiti a degli «eseguibili», ovvero i programmi e gli script, sono indicati in modo evidenziato, esattamente come si utilizzano nel sistema operativo, senza cambiamenti nella collezione alfabetica delle lettere maiuscole e minuscole. Quando però il programma riveste un'importanza particolare, può assumere una denominazione diversa da quella che si usa nel nome del file eseguibile, oppure semplicemente si può decidere di trattarlo come qualcosa di più importante.

Per fare un esempio pratico, quando si parla di shell si fa riferimento alla shell Bash, alla shell Korn, alla shell C,... mentre l'eseguibile vero e proprio potrebbe essere '**`bash`**', '**`ksh`**', '**`csh`**',... Lo stesso vale per i programmi che meritano questa attenzione anche se il loro nome (verbale) non cambia.

In generale, il nome di un programma applicativo, di un pacchetto o di altre situazioni analoghe, viene indicato con l'iniziale maiuscola, salvo eccezioni che possono derivare dall'uso acquisito in una qualche forma differente, escludendo a ogni modo l'uso di sole lettere minuscole.

> Il nome di un programma eseguibile va annotato in forma dattilografica, esattamente come deve essere scritto per avviarlo, ovvero come indicato nel file system. Nell'ambito dello stile dell'opera, quando si scrive il nome di un programma senza voler fare riferimento al file eseguibile, il nome in questione **non** può essere annotato usando solo lettere minuscole, anche se l'autore originale fa così.

La tabella 14.2 elenca alcune delle scelte di stile nell'uso dei nomi dei programmi distinguendo tra «eseguibile» e qualcosa di diverso: applicativo, pacchetto, servizio, sistema e simili, riferite a forme che costituiscono un'eccezione rispetto alla regola generale.

Tabella 14.2. Stile nell'uso dei nomi dei programmi distinguendo tra «eseguibile» e «applicativo», limitatamente ad alcune eccezioni.

| Eseguibile | Applicativo, pacchetto, servizio, sistema,... |
|---|---|
| '**`lilo`**' | LILO |
| '**`*getty`**' | Getty |
| '**`getty`**', '**`uugetty`**' | Getty_ps |
| '**`mgetty`**' | Mgetty+Sendfax |
| '**`bash`**' | shell Bash |
| '**`csh`**' | shell C |
| '**`ksh`**' | shell Korn |
| '**`sh`**' | shell Bourne |
| '**`init`**' | Procedura di inizializzazione del sistema, Init |
| '**`cron`**' (demone) | Cron (sistema) |
| '**`inetd`**' | supervisore dei servizi di rete |
| '**`tcpd`**' | TCP wrapper |
| '**`portmap`**' | Portmapper |
| '**`named`**' | BIND (pacchetto) |
| '**`telnet`**' | Telnet (programma)<br>TELNET (protocollo o servizio) |
| '**`finger`**' | Finger (servizio) |
| '**`sendmail`**' | Sendmail |
| '**`mail`**' | Mailx |
| '**`ex`**' | EX |
| '**`vi`**' | VI |
| '**`joe`**' | Joe |
| '**`m4`**' | M4 |

| Eseguibile | Applicativo, pacchetto, servizio, sistema,... |
|---|---|
| ‘**mc**’ | Midnight Commander |
| ‘**nsgmls**’ | SP |
| ‘**sgmlspl**’ | SGMLSpm |
| ‘**gs**’ | Ghostscript |
| ‘**bmv**’ | BMV |
| ‘**ghostview**’ | Ghostview |
| ‘**gv**’ | GV |
| ‘**xpaint**’ | XPaint |
| ‘**ee**’, ‘**eeyes**’ | Electric Eyes |
| ‘**xfm**’ | XFM |
| ‘**tcd**’, ‘**gtcd**’ | TCD |
| ‘**xcdroast**’ | X-CD-Roast |

## 14.5 Descrizione degli acronimi

Gli acronimi non sono sempre ottenuti con le sole iniziali delle parole che compongono il nome di qualcosa; inoltre, non c’è alcuna necessità pratica nell’evidenziare la corrispondenza tra le lettere usate e la frase corrispondente. In questo senso, la descrizione degli acronimi che si fa con l’elemento ‘**dacronym**’ ha un aspetto uniforme: l’iniziale maiuscola e il resto del testo in minuscolo, tranne nel caso in cui si tratti di termini che rappresentano dei nomi importanti o degli altri acronimi, oppure quando la lingua di origine impone l’uso della maiuscola. Seguono alcuni esempi:

| Acronimo | Descrizione completa | Annotazioni |
|---|---|---|
| MTA | *Mail transfer agent* | |
| XML | *Extensible markup language* | |
| ORF | *Österreichischer Rundfunk* | Nella lingua tedesca i sostantivi hanno l’iniziale maiuscola. |
| MIME | *Multipurpose Internet mail extentions* | Il nome che contiene (Internet) si scrive comunemente con l’iniziale maiuscola. |

## 14.6 Indice analitico

Il problema della costruzione di un indice analitico è già trattato nel capitolo sullo stile letterario in generale. All’interno dell’opera *Appunti di informatica libera* ci sono delle particolarità che è bene precisare.

In particolare, l’indice analitico realizzato con il sistema di composizione di *Appunti di informatica libera* consente l’uso di un carattere dattilografico attraverso l’uso dell’elemento ‘**code**’, oppure assegnando il valore ‘**code**’ all’attributo ‘**emph**’:

```
<indexentry>Perl: <code>print</code></indexentry>
```

```
<indexentry><code>/etc/profile</code></indexentry>
```

- I termini inseriti nell’indice analitico vanno scritti usando lettere minuscole, a meno che si tratti di nomi particolari che vanno sempre scritti in un modo prestabilito.
  - La descrizione di un acronimo, inserita per esteso, si scrive con le stesse regole usate per l’elemento ‘**dacronym**’, per cui l’iniziale è maiuscola.

  - Il nome di un applicativo, di un pacchetto, di un servizio, di un sistema e simili, va scritto nello stesso modo usato nel testo normale, senza cambiare lo stato delle lettere maiuscole e minuscole.
  - Il nome di file e directory va scritto esattamente come appare nel sistema operativo, utilizzando un carattere dattilografico, tenendo conto che i file eseguibili vanno indicati senza percorso, mentre gli altri dovrebbero contenerlo.
  - Il nome delle variabili di ambiente va scritto esattamente come appare nel sistema operativo (generalmente si tratta di nomi scritti con lettere maiuscole), usando un carattere dattilografico, lasciando il dollaro come prefisso.
  - Quando si inserisce il nome di un applicativo che possiede un eseguibile con lo stesso nome, non si annota anche il nome dell'eseguibile. In pratica, se si inserisce la voce «Pippo» senza enfatizzazione, non si annota anche la voce «pippo», corrispondente all'eseguibile omonimo, in modo dattilografico; al massimo, si inserisce un'altra volta la stessa voce «Pippo». Infatti, chi cerca notizie sul programma Pippo, o sull'eseguibile '**`pippo`**', si troverebbe in difficoltà nello scegliere tra l'una e l'altra voce. Quando invece un applicativo si articola in programmi eseguibili differenti, è sensato annotare sia il nome dell'applicativo, sia i nomi degli eseguibili che vengono descritti in modo particolare.
  - Quando la voce «Pippo» è comunque una cosa diversa da «pippo», le due voci vanno annotate esattamente e separatamente. Per esempio, si può fare riferimento al protocollo FTP e poi al programma eseguibile '**`ftp`**'. Il lettore può sentirsi confuso dalla distinzione, ma in tal caso è necessaria.

- Si utilizza il singolare, salvo eccezioni dovute al fatto che il termine al singolare possa intendersi come una cosa differente da ciò che si vuole realmente.

- La prima parola dovrebbe essere un sostantivo, o comunque è necessario sostantivare l'inizio della voce da inserire nell'indice analitico.

- Non si inizia una voce dell'indice analitico con un verbo; nel caso si può sostantivare il verbo. Per esempio, al posto di «salvare i dati» si può inserire la voce «salvataggio dei dati».

- Il sistema di composizione non consente l'indicazione di sottoclassificazioni nell'indice analitico, per cui si usa la tecnica seguente:

  | ***voce*** : ***sottoclassificazione*** |
  |---|

  Questo fatto implica che i due punti vadano usati solo per questo scopo nelle voci dell'indice analitico; inoltre, diventa inopportuno l'inserimento di una sottoclassificazione ulteriore.

  - Una sottoclassificazione non è sottoposta all'obbligo di essere formulata usando il singolare; tuttavia, in caso di conflitto, si deve preferire la forma al singolare.
  - Una sottoclassificazione inizia con un sostantivo, così come iniziano le voci normali. Per esempio, «salvataggio: recuperare i dati» va sostituito con «salvataggio: recupero dei dati».
  - Non si usa il trattino per indicare una sottoclassificazione. Per esempio, «salvataggio -- recupero dei dati» va sostituto con «salvataggio: recupero dei dati».

- Quando si inserisce una voce in una sezione, non si inserisce nuovamente nelle sottosezioni relative. In pratica, se si inserisce la voce «Pippo» in corrispondenza dell'inizio di un capitolo, non si inserisce nuovamente la stessa voce in altre sezioni inferiori dello stesso capitolo.

- Le voci dell'indice analitico vanno inserite in riferimento alle sezioni opportune. Per esempio, la parola «file» potrebbe trovarsi in quasi tutte le pagine di un testo di informatica, mentre dovrebbe essere fatto un richiamo solo a quelle sezioni in cui si spiega di cosa si tratta (ammesso che ci sia).

I riferimenti per la generazione dell'indice analitico vanno posti preferibilmente nel titolo della sezione a cui fanno riferimento, come nell'esempio seguente:

```
<H3>
Copie di sicurezza
<indexentry>salvataggio: copia di sicurezza</indexentry>
<indexentry>salvataggio: recupero dei dati</indexentry>
</H3>
```

Come si vede, viene indicato prima il titolo e subito dopo l'elenco dei riferimenti da inserire nell'indice, che riguardano la sezione.

Inserendo le voci dell'indice analitico nell'ambito del titolo di una sezione, si comprende che non abbia senso ripetere la stessa voce nelle sottosezioni relative.

## 14.7 Enfatizzazioni e uso degli elementi «special»

La gestione corretta delle «enfatizzazioni» è sempre un problema serio di coerenza, soprattutto se si considera il fatto che l'enfatizzazione non implica solo la composizione finale con un aspetto particolare, ma anche la classificazione dell'oggetto per qualche fine. In particolare, l'elemento '**`special`**' non genera alcuna enfatizzazione, ma serve a dare una classificazione al termine inserito, per qualche ragione. L'opera *Appunti di informatica libera* usa le convenzioni che vengono sintetizzate in questa sezione.

- `<samp>`***stringa***`</samp>`

  Si usa all'interno di un testo normale per delimitare delle stringhe che hanno un valore letterale e si riferiscono in qualche modo a un'informazione tecnica. In particolare, si indicano in questo modo:

  – i nomi degli eseguibili;
  – gli esempi di opzioni di una riga di comando;
  – i nomi delle variabili di ambiente (senza il dollaro iniziale);
  – i nomi di elementi SGML (compreso XML e altre applicazioni);
  – gli esempi di istruzioni, comandi e direttive di qualunque tipo;
  – tutte le informazioni tecniche letterali che non ricadono in situazioni differenti.

- `<code>`***nome***`</code>`

  Si tratta di una forma di enfatizzazione molto simile a quella dell'elemento '**`samp`**', riservata a situazioni particolari:

  – può essere usata per ottenere un carattere dattilografico nelle voci dell'indice analitico;
  – l'elemento '**`code`**' può essere usato come **unico** elemento contenuto all'interno di '**`dt`**', quando in condizioni normali questo sarebbe stato rappresentato con l'elemento '**`samp`**';

  - l'elemento '**code**' può essere usato come **unico** elemento contenuto all'interno di '**faqh3**', quando in condizioni normali questo sarebbe stato rappresentato con l'elemento '**samp**'.

- `<indexentry>…<code>`***stringa***`</code>…</indexentry>`

  Nell'ambito delle voci dell'indice analitico, si può usare solo l'elemento '**code**' per indicare qualunque cosa che debba essere annotata in modo dattilografico. In pratica, tutto ciò che nel testo normale dovrebbe essere inserito usando elementi che generano in qualche modo una composizione dattilografica va scritto nell'indice analitico usando l'elemento '**code**'.

- `<file>`***file***`</file>`

  Nel testo normale, i nomi di file e directory, con o senza percorsi, vanno inseriti nell'elemento '**file**'. In generale, il nome di un file o di una directory dovrebbe sempre contenere l'informazione del percorso, salvo che si tratti implicitamente della directory corrente, oppure che non si possa stabilire una posizione precisa.

  > Si usa la convenzione delle shell derivate da quella di Bourne, per cui il simbolo '`~/`' rappresenta la directory personale dell'utente che sta usando il sistema, mentre '`~`***utente***`/`' rappresenta la directory personale dell'utente indicato.

  In un percorso del genere si può inserire l'elemento '**var**', per descrivere una parte variabile dello stesso; inoltre è ammesso l'uso di caratteri jolly elementari, ovvero asterisco e punto interrogativo, per fare riferimento a più file.

  I nomi delle directory terminano sempre con la barra finale: '`/`' o '`\`' a seconda del sistema operativo a cui si fa riferimento.

  Quando si vuole fare riferimento a un file contenente un documento che dovrebbe essere raggiungibile in ogni sistema che abbia installato un certo applicativo, si può usare eventualmente l'elemento '**uri**', indicando un URI di tipo '**file:**', allo scopo di consentire l'accesso ipertestuale al file stesso. Naturalmente, ciò ha senso se l'URI che si indica è valido; quindi non è il caso di indicare caratteri jolly in un indirizzo del genere.

- `<var>`***metavariabile***`</var>`

  L'elemento '**var**' serve a delimitare una metavariabile, ovvero qualcosa che **descrive** ciò che va sostituito al suo posto. Non si indicano con questo elemento altri tipi di variabili, come potrebbero essere le variabili di ambiente o quelle di un programma scritto con un certo linguaggio. In tal caso, si userebbe piuttosto l'elemento '**samp**'.

  L'elemento '**var**' va usato prevalentemente all'interno dell'elemento '**syntax**', nei modelli sintattici, ma può essere usato utilmente anche dentro un elemento '**samp**', quando una parte della stringa non è fissa, così come in un elemento '**file**', per lo stesso motivo.

  Eccezionalmente, si può indicare un comando con l'inserzione di un elemento '**var**' all'interno del testo da digitare, ovvero l'elemento '**type**'. Tuttavia, in condizioni normali, si preferisce fare questo in un elemento '**syntax**', se il contesto lo consente.

  È consentita l'inserzione dell'elemento '**var**' anche all'interno di un elemento '**pre**', quando non è opportuno l'uso di un elemento '**syntax**' al suo posto.

  > Il nome di una metavariabile dovrebbe descrivere ciò che rappresenta, mentre non deve essere un esempio del contenuto.

Per evitare confusione, il nome va scritto usando possibilmente lettere minuscole, dove le varie parti possono essere separate da un trattino basso, come nel caso di ***mia_variabile***. Naturalmente si possono usare anche i numeri, purché sia chiaro che servono solo a individuare la metavariabile, come nel caso di ***nome_1***, ***nome_2***,... ***nome_n***. È da escludere l'uso di altri segni, perché creerebbero confusione, dal momento che i nomi delle variabili non appaiono delimitati. Se possibile è meglio evitare l'uso dell'apostrofo.

Se possibile, è meglio comporre il nome delle metavariabili usando termini normali (non abbreviati o fusi assieme), in modo da non doverli inserire inutilmente nel vocabolario del controllo ortografico.

- `<dfn>`***definizione***`</dfn>`

  L'elemento '**`dfn`**' serve a delimitare una definizione, ovvero un termine che viene introdotto in riferimento a un contesto particolare. Va usato solo quando viene introdotto e non ha altro scopo che quello di generare una forma di evidenziamento uniforme.

  Lo stesso termine può apparire in contesti differenti e con un significato diverso; pertanto, l'uso dell'elemento '**`dfn`**' vale in quanto riferito al contesto particolare a cui appartiene la parola evidenziata.

  In generale, è bene evitare la proliferazione di evidenziamenti del genere, che vanno limitati alle situazioni in cui si vuole cogliere l'attenzione del lettore.

- `<strdfn>`***definizione_straniera***`</strdfn>`

  L'elemento '**`strdfn`**' serve a delimitare un termine o una definizione in lingua straniera, che non si intende utilizzare nel testo come terminologia normale, ma solo per spiegare, eventualmente, a cosa si sta facendo riferimento.

- `<em>`***testo***`</em>`
  `<strong>`***testo***`</strong>`
  `<small>`***testo***`</small>`
  `<big>`***testo***`</small>`

  Le forme di evidenziamento generico vanno usate con molta parsimonia, perché non esiste una regola generale per il loro utilizzo. In particolare, un carattere ingrandito ottenuto con l'elemento '**`big`**' è utile nella realizzazione di presentazioni (lucidi per lavagna luminosa).

- `<bibref>`***titolo***`</bibref>`

  Si usa l'elemento '**`bibref`**', nel testo normale, per delimitare il titolo di un documento o di un'opera di qualunque tipo.

- `<kerneloption>`***opzione_del_kernel***`</kerneloption>`

  Si usa l'elemento '**`kerneloption`**', nel testo normale, per delimitare la voce corrispondente a un'opzione del kernel.

- `<dacronym>`***descrizione_acronimo***`</dacronym>`

  Si usa l'elemento '**`dacronym`**', nel testo normale, per delimitare la descrizione di un acronimo.

- `<acronym>`***acronimo***`</acronym>`

  Questo elemento dovrebbe servire per delimitare un acronimo, secondo la logica del sistema di composizione, ma attualmente gli acronimi non vengono delimitati in alcun modo.

- `<kbd>`***combinazione_tasti***`</kbd>`

  L'elemento '**`kbd`**' viene usato per indicare tasti (della tastiera) o combinazioni di tasti da premere. I nomi dei tasti vanno indicati come previsto (tabella 15.3, nel capitolo 15)

e le combinazioni si ottengono inserendo il segno '**+**' tra i vari nomi o tra i simboli corrispondenti.

Nelle tabelle, quando si elencano tasti e combinazioni di tasti, si può fare a meno di questa forma di enfatizzazione.

- `<button>`***pulsante_grafico***`</button>`

  L'elemento '**`button`**' viene usato per indicare il nome di pulsanti grafici, anche in presenza di terminali a caratteri, che si selezionano attraverso un cursore o un puntatore grafico. Non si usa questo elemento per indicare l'uso della tastiera normale.

- `<menuitem>`***voce_di_menù***`</menuitem>`

  Si delimitano in questo modo le voci di un programma grafico o di uno per terminali a caratteri che abbia un comportamento simile a quelli grafici, che siano riconducibili a scelte di un menù di funzioni. Rientrano in questa situazione i menù a tendina, i nomi delle etichette dei lembi di una sistema di cartelle, oppure il nome di un tipo di selezione che non sia riconducibile a un pulsante.

  Questo elemento può essere usato anche per evidenziare le voci che rappresentano un tipo di casella di selezione, oppure le etichette dei campi in cui deve essere inserito qualche tipo di informazione.

- `<asciicode>`***nome_ascii***`</asciicode>`

  Si delimitano in questo modo i nomi di caratteri speciali ASCII, che secondo la tradizione sono rappresentati da abbreviazioni con lettere maiuscole. La tabella 14.4 elenca tutti i caratteri che possono essere rappresentati in questo modo, mostrando anche il risultato dell'utilizzo dell'elemento.

Tabella 14.4. Elenco dei caratteri speciali che si possono inserire nell'elemento '**`asciicode`**'.

| Binario | Esadecimale | Ottale | Decimale | Carattere |
|---|---|---|---|---|
| $00000000_2$ | $00_{16}$ | $000_8$ | $000_{10}$ | *<NUL>* |
| $00000001_2$ | $01_{16}$ | $001_8$ | $001_{10}$ | *<SOH>* |
| $00000010_2$ | $02_{16}$ | $002_8$ | $002_{10}$ | *<STX>* |
| $00000011_2$ | $03_{16}$ | $003_8$ | $003_{10}$ | *<ETX>* |
| $00000100_2$ | $04_{16}$ | $004_8$ | $004_{10}$ | *<EOT>* |
| $00000101_2$ | $05_{16}$ | $005_8$ | $005_{10}$ | *<ENQ>* |
| $00000110_2$ | $06_{16}$ | $006_8$ | $006_{10}$ | *<ACK>* |
| $00000111_2$ | $07_{16}$ | $007_8$ | $007_{10}$ | *<BEL>* |
| $00001000_2$ | $08_{16}$ | $010_8$ | $008_{10}$ | *<BS>* |
| $00001001_2$ | $09_{16}$ | $011_8$ | $009_{10}$ | *<HT>* |
| $00001010_2$ | $0A_{16}$ | $012_8$ | $010_{10}$ | *<LF>* |
| $00001011_2$ | $0B_{16}$ | $013_8$ | $011_{10}$ | *<VT>* |
| $00001100_2$ | $0C_{16}$ | $014_8$ | $012_{10}$ | *<FF>* |
| $00001101_2$ | $0D_{16}$ | $015_8$ | $013_{10}$ | *<CR>* |
| $00001110_2$ | $0E_{16}$ | $016_8$ | $014_{10}$ | *<SO>* |
| $00001111_2$ | $0F_{16}$ | $017_8$ | $015_{10}$ | *<SI>* |
| $00010000_2$ | $10_{16}$ | $020_8$ | $016_{10}$ | *<DLE>* |
| $00010001_2$ | $11_{16}$ | $021_8$ | $017_{10}$ | *<DC1>* |
| $00010010_2$ | $12_{16}$ | $022_8$ | $018_{10}$ | *<DC2>* |
| $00010011_2$ | $13_{16}$ | $023_8$ | $019_{10}$ | *<DC3>* |
| $00010100_2$ | $14_{16}$ | $024_8$ | $020_{10}$ | *<DC4>* |
| $00010101_2$ | $15_{16}$ | $025_8$ | $021_{10}$ | *<NAK>* |
| $00010110_2$ | $16_{16}$ | $026_8$ | $022_{10}$ | *<SYN>* |
| $00010111_2$ | $17_{16}$ | $027_8$ | $023_{10}$ | *<ETB>* |
| $00011000_2$ | $18_{16}$ | $030_8$ | $024_{10}$ | *<CAN>* |
| $00011001_2$ | $19_{16}$ | $031_8$ | $025_{10}$ | *<EM>* |

| Binario | Esadecimale | Ottale | Decimale | Carattere |
|---|---|---|---|---|
| $00011010_2$ | $1A_{16}$ | $032_8$ | $026_{10}$ | *<SUB>* |
| $00011011_2$ | $1B_{16}$ | $033_8$ | $027_{10}$ | *<ESC>* |
| $00011100_2$ | $1C_{16}$ | $034_8$ | $028_{10}$ | *<FS>* |
| $00011101_2$ | $1D_{16}$ | $035_8$ | $029_{10}$ | *<GS>* |
| $00011110_2$ | $1E_{16}$ | $036_8$ | $030_{10}$ | *<RS>* |
| $00011111_2$ | $1F_{16}$ | $037_8$ | $031_{10}$ | *<US>* |
| $00100000_2$ | $20_{16}$ | $040_8$ | $032_{10}$ | *<SP>* |
| $01111111_2$ | $7F_{16}$ | $177_8$ | $127_{10}$ | *<DEL>* |

La sequenza di più caratteri del genere si ottiene semplicemente mettendo a contatto più elementi '**asciicode**', come per esempio nel caso di *<CR><LF>*.

- `<uristr>`***uri_non_ipertestuale***`</uristr>`

  L'elemento '**uristr**' si affianca all'elemento '**uri**', con lo scopo di rappresentare degli indirizzi URI per i quali non si vuole realizzare un riferimento ipertestuale. Ciò si rende necessario quando si scrive un indirizzo di fantasia o un indirizzo che si vuole conservare pur non essendo più valido. Si usa questo elemento anche quando si tratta di nomi di dominio, senza l'indicazione di una risorsa precisa.

- `<special special="name">`***nome***`</special>`

  Serve a delimitare, senza evidenziare, un nome. Si ottiene l'elenco dei nomi usati con l'inserzione del marcatore '**<printindex index="name">**', allo scopo di verificare di avere usato sempre lo stesso modo. Ciò si rende necessario per mantenere coerenza nell'uso delle maiuscole e di altri simboli eventuali (come il trattino normale, il trattino basso e altre inserzioni che potrebbero fare parte del nome in qualche modo).

  In generale si utilizza questo elemento per i nomi dei programmi e dei pacchetti di programmi. Per tutte le altre situazioni conviene controllare se nel sorgente è già stato usato un elemento del genere per il nome che si inserisce.

- `<special special="ttid">`***termine***`</special>`

  Serve a delimitare, senza evidenziare, un termine particolare, espresso in italiano, per il quale si vuole avere un controllo. In generale ciò serve a seguire delle definizioni che non sono comuni ed è bene mantenere coerenti, per non confondere il lettore. Un'altra ragione per questo utilizzo è quello di facilitare la ricerca di tali definizioni nel momento in cui si decidesse di sostituirle con altre. Ciò si rende necessario perché un termine può avere quel certo significato speciale solo in un contesto particolare; pertanto, solo in questi casi va delimitato così.

  I termini delimitati in questo modo sono evidenziati nel capitolo 15 con l'aggiunta di un asterisco.

- `<special special="ttsc">`***termine***`</special>`

  Serve a delimitare, senza evidenziare, un termine particolare, espresso in inglese (o in un'altra lingua straniera), che per qualche ragione non sia traducibile, ma che non sia ancora stato acquisito completamente nella lingua italiana. L'elenco di questi termini si trova nella tabella 15.2 (capitolo 15).

## 14.8 Rappresentazione del contenuto di file e dei flussi standard

In generale, il contenuto di un file o quanto emesso da un programma attraverso standard output e standard error, viene rappresentato in un elemento per il testo preformattato. Tuttavia, si manifestano dei problemi estetici, dovuti alla suddivisione del testo in pagine e al riconoscimento del contesto.

Per controllare la possibilità o meno di spezzare il testo tra più pagine, si inserisce l'elemento che lo contiene in un listato (l'elemento '**`listing`**') fisso, che, a seconda di ciò che si preferisce, possa essere spezzato o meno:

```
<listing pos="fixed" split="0">
...
...
</listing>
```

In questo caso, evidentemente, si tratta di un listato che non si può spezzare; la scelta se mantenere unito o consentire la divisione in più pagine dipende naturalmente dalla lunghezza del testo.

Per quanto riguarda l'uso di linee e bordi di separazione, all'inizio del sorgente sono dichiarate alcune macro per la definizione dello stile, in modo da consentire in un secondo momento di cambiare l'aspetto generale. Si distinguono i casi seguenti, dimostrati da esempi:

- listato riferito al contenuto di un file su disco (che può essere anche uno script);

```
<listing sep="&STYLE.LISTING.SEP.FILE.CONTENT;">
<verbatimpre border="&STYLE.PRE.BORDER.FILE.CONTENT;">
...
...
</verbatimpre>
</listing>
```

```
<listing sep="&STYLE.LISTING.SEP.FILE.CONTENT;">
<pre border="&STYLE.PRE.BORDER.FILE.CONTENT;">
...
...
</pre>
</listing>
```

- listato riferito a quanto emesso attraverso lo standard output o lo standard error;

```
<listing sep="&STYLE.LISTING.SEP.OUTPUT.COMMAND;">
<verbatimpre border="&STYLE.PRE.BORDER.OUTPUT.COMMAND;">
...
...
</verbatimpre>
</listing>
```

```
<listing sep="&STYLE.LISTING.SEP.OUTPUT.COMMAND;">
<pre border="&STYLE.PRE.BORDER.OUTPUT.COMMAND;">
...
...
</pre>
</listing>
```

- listato riferito a quanto appare sullo schermo a seguito dell'utilizzo di un programma interattivo;

```
<listing sep="&STYLE.LISTING.SEP.OUTPUT.SCREEN;">
<verbatimpre border="&STYLE.PRE.BORDER.OUTPUT.SCREEN;">
...
...
</verbatimpre>
</listing>
```

```
<listing sep="&STYLE.LISTING.SEP.OUTPUT.SCREEN;">
<pre border="&STYLE.PRE.BORDER.OUTPUT.SCREEN;">
...
...
</pre>
</listing>
```

## 14.9 Altri problemi di coerenza nell'uso degli elementi SGML

La coerenza in ciò che poi si traduce in forme di enfatizzazione del testo è la cosa più importante da definire e anche la più difficile da mantenere. Tuttavia, ci sono altre considerazioni da fare su elementi che potrebbero sembrare più ovvi.

- I titoli della serie '**tomeheading**', '**h0**', '**h1**', '**h2**', '**h3**', '**h4**', '**slideh1**', '**sheeth1**' e '**faqh2**', vanno scritti senza inserire enfatizzazioni di alcun genere. Tuttavia, si possono e si devono inserire gli elementi '**special**'. In caso di necessità, si può delimitare qualche termine particolare solo usando le parentesi angolari uncinate standard.

  Come si vede, a questa regola fa eccezione '**faqh3**' che invece può contenere le enfatizzazioni comuni di un testo normale.

- Le tabelle vanno realizzate nel modo più semplice possibile, cercando di evitare contorsioni, allo scopo di facilitare la lettura anche a un utente che si limiti a scorrere il documento in forma di testo puro e semplice. Solo eccezionalmente è utile la realizzazione di tabelle HTML, racchiuse nell'elemento '**html**', per rappresentare schemi particolari, come nel caso delle schede riepilogative.

- Quando una figura può essere realizzata facilmente utilizzando semplicemente caratteri ASCII, conviene evitare la grafica, per consentire la visualizzazione della stessa anche in forma di testo puro. Si ottiene facilmente una figura del genere con l'elemento '**asciiart**', oppure anche solo con '**verbatimpre**'.

- A seconda dei tipi di composizione si possono avere pagine che hanno altezze molto diverse. Quando si realizza una tabella o una figura, occorre verificare che la composizione A4 normale avvenga correttamente; di conseguenza saranno corrette anche le altre forme.

## 14.10 Sezioni marcate per le annotazioni

Vengono usate delle sezioni marcate per inserire delle annotazioni da ottenere solo nella stampa di bozze. Queste sezioni marcate fanno riferimento all'entità parametrica '**ANNOTAZIONI**'. Di solito si fanno queste annotazioni utilizzando delle note a piè pagina, usando l'elemento '**blockfootnote**', che permette di non intaccare un blocco di testo normale. Si distinguono due tipi di segnalazioni: un'informazione da ricordare e un problema non risolto, da sistemare in un secondo momento. Si osservino i due esempi seguenti:

```
<![%ANNOTAZIONI;[

    <blockfootnote><strong>ATTENZIONE</strong> --- questa notizia
    proviene da una ricerca fatta... così e così...</blockfootnote>

]]><!--%ANNOTAZIONI;-->
```

```
<![%ANNOTAZIONI;[

    <blockfootnote><strong>SISTEMARE</strong> --- manca da analizzare
    la questione relativa alla...</blockfootnote>

]]><!--%ANNOTAZIONI;-->
```

A differenza delle altre sezioni marcate normali, queste vengono incolonnate come i blocchi di testo:

```
<ul>
<li>

    <p>Bla bla bla bla...</p>

    <![%ANNOTAZIONI;[

        <blockfootnote><strong>SISTEMARE</strong> --- manca da analizzare
        la questione relativa alla...</blockfootnote>

    ]]><!--%ANNOTAZIONI;-->

    <p>Bla bla bla bla...</p>

</li>
<li>

    ...

</li>
</ul>
```

*Capitolo* 15

# Glossario stilistico di «Appunti di informatica libera»

Come accennato nell'introduzione dell'opera, quando si scrivono documenti a carattere tecnico in lingua italiana, è difficile essere comprensibili, coerenti e anche corretti secondo le regole della lingua. Inoltre non si può nemmeno contare sulla presenza di una qualche autorità in grado di dare risposte a dei quesiti sul modo giusto di definire o di esprimere qualcosa.

Nel capitolo 3 sono raccolti dei punti di riferimento, tuttavia resta aperto il problema della terminologia da adoperare. Attualmente, esiste la lista `it@li.org` che si occupa di discutere i problemi legati alle traduzioni di documenti come HOWTO, pagine di manuale e messaggi dei programmi GNU. La traduzione è una cosa differente dallo scrivere qualcosa di nuovo in italiano, ma comunque, la sensibilità e le scelte di ognuno possono essere diverse.

In questo capitolo si raccolgono alcune annotazioni sulle forme stilistiche ed espressive usate o che potrebbero essere usate in futuro in questa opera (nel tempo sono cambiate molte cose in questo documento e dovrebbero cambiarne ancora molte altre).

Sono sempre graditi i commenti riferiti al contenuto di questo capitolo e a tutto il resto dell'opera.

Alla fine del capitolo appare un indice analitico delle voci che sono state trattate qui. Ciò per facilitarne la ricerca, dal momento che i termini in questione appaiono secondo un certo ordine «logico», che non è quello alfabetico.

Nelle annotazioni delle sezioni seguenti, appaiono alcune sigle che hanno un significato molto semplice:

- *m.* -- maschile;
- *f.* -- femminile;
- *s.* -- singolare;
- *inv.* -- invariato al plurale;
- *agg.* -- aggettivo.

Il capitolo è organizzato secondo la struttura seguente:

## 15.1 Termini tecnici particolari

Sono considerati acquisiti in italiano i termini tecnici elencati nella tabella 15.1. In quanto tali, sono indicati nel testo dell'opera e nel sorgente stesso senza enfatizzazioni tipografiche.

Tabella 15.1. Elenco dei termini tecnici considerati acquisiti nel linguaggio.

| Termine | Annotazioni |
|---|---|
| bit | s. m. inv. |
| byte | s. m. inv. |
| computer | s. m. inv. -- meglio «elaboratore» |
| console | s. f. inv. |
| directory | s. f. inv. |
| sottodirectory | s. f. inv. |
| file | s. m. inv. |
| hardware | s. m. inv. |
| input | s. m. inv. |
| mixer | s. m. inv. |
| modem | s. m. inv. |
| monitor | s. m. inv. |
| mouse | s. m. inv. |
| output | s. m. inv. |
| routine | s. f. inv. |
| subroutine | s. f. inv. |
| software | s. m. inv. |
| timer | s. m. inv. |
| zoom | s. m. inv. |

Inoltre, i termini che ormai sembrano far parte del linguaggio tecnico italiano in modo irrimediabile, sono annotati nella tabella 15.2. Anche questi appaiono nel testo dell'opera senza enfatizzazioni tipografiche, ma nel sorgente sono delimitati in modo da poter essere riconoscibili, attraverso la forma:

```
<special special="ttsc">termine</special>
```

Tabella 15.2. Elenco dei termini tecnici apparentemente consolidati in italiano, oppure che risultano intraducibili per qualche motivo. Nella tabella si annotano anche i termini che sarebbero traducibili, ma che hanno qualche particolarità se usati invariati in italiano.

| Termine | Annotazioni |
|---|---|
| anycast | agg. -- IPv6 |
| applet | s. f. inv. -- «applicazioncina» |
| array | s. m. inv. |
| bridge | s. m. inv. |
| gateway | s. m. inv. |
| router | s. m. inv. |
| broadcast | agg. |
| bus | s. m. inv. |
| cast | s. m. inv. |
| crontab | s. m. inv. -- file di Cron |
| dot-clock | s. m. inv. |
| driver | s. m. inv. -- meglio «gestore» |
| escape | s. m. inv. / agg. |
| feed | s. m. inv. -- Usenet |
| file di lock | s. m. inv. |
| file system | s. m. inv. -- meglio evitare «filesystem» |
| firewall | s. m. inv. |
| firmware | s. m. inv. |
| fuzzy | agg. -- logica |
| hash | s. m. inv. -- array associativi di Perl |
| inode | s. m. inv. |
| job | s. m. inv. |
| join | s. m. inv. -- basi di dati |
| joystick | s. m. inv. |
| kernel | s. m. inv. |
| led | s. m. inv. -- i diodi led |
| link | s. m. inv. -- compilazione |
| linker | s. m. inv. -- compilazione |
| link-local | agg. -- IPv6 |
| magic number | s. m. inv. |
| memoria cache | s. f. inv. |
| multicast | agg. |
| node-local | agg. -- IPv6 |
| news | s. f. inv. |
| nice | agg. -- valore nice |
| organization-local | agg. |
| *password* | s. f. inv. -- qui si preferisce parola d'ordine |
| ping | s. m. inv. -- «fare il ping» |
| pipe | s. f. inv. |
| pipeline | s. f. inv. |
| pixel | s. m. inv. |
| proxy | s. m. inv. -- se il contesto non è specifico, meglio parafrasare |
| record | s. m. inv. |
| script | s. m. inv. |
| shadow | s. f. inv. -- password shadow |

| Termine | Annotazioni |
|---|---|
| shell | s. f. inv. |
| subshell | s. f. inv. |
| site-local | agg. -- IPv6 |
| socket | s. m. inv. |
| stack | s. m. inv. -- quello di un processo, per salvare i registri |
| standard input | s. m. inv. |
| standard output | s. m. inv. |
| standard error | s. m. inv. |
| task | s. m. inv. -- se possibile, meglio parafrasare |
| unicast | agg. -- IPv6 |
| *utility* | s. f. inv. -- meglio «programma di servizio» o al limite «programma di utilità» |

Le regole per la definizione del genere maschile o femminile per un termine tecnico proveniente dalla lingua inglese, che viene usato così com'è in italiano, sono molto vaghe. Inoltre, i termini inglesi che vengono incorporati nell'italiano vanno usati generalmente al singolare, anche quando esprimono quantità multiple.

### 15.1.1 Annotazioni sui termini tecnici ritenuti «intraducibili»

- array

  Il termine array rappresenta una struttura di dati particolare, mentre i termini «vettore» e «matrice» sono specifici della matematica.

- bridge; router; gateway

  Queste parole servono a definire in modo preciso e standard il ruolo di uno di quei nodi di rete che permettono un attraversamento tra una sottorete e un'altra.

- directory

  Il termine directory è stato tradotto in passato in vari modi poco soddisfacenti. Il concetto più elegante che si possa abbinare alla directory è quello di «cartella», che però è conveniente solo in presenza di un sistema operativo prevalentemente grafico.

- feed (Usenet)

  È difficile trovare una traduzione accettabile per esprimere il feed degli articoli di Usenet. Eventualmente si potrebbe parlare di «propagazione» degli articoli, quando il contesto lo consente, dal momento che non è proprio la stessa cosa.

- file di lock

  Il file di lock è un file che indica il blocco di un qualche tipo di risorsa (blocco perché la risorsa è impegnata in qualche modo e non è consentito l'accesso da parte di altri processi). È difficile tradurre questa forma perché l'espressione è radicata molto bene negli ambienti tecnici e fino a questo momento non è venuta fuori alcuna alternativa altrettanto comprensibile: «file di blocco», che sarebbe la traduzione corretta, rischia di fare pensare ad altro, ingannando il lettore.

  Se c'è la possibilità di parafrasare, si potrebbe fare riferimento a un «file per il controllo dell'accesso», oppure a un «file di protezione» contro gli accessi concorrenziali a una risorsa data. Se poi non è necessario fare riferimento all'uso di questo file, ci si può riferire direttamente al fatto che si impedisce l'accesso da parte di altri processi, oppure che si protegge qualcosa contro gli accessi concorrenziali.

Quando si parla di un blocco attraverso funzioni del sistema operativo, non è il caso di usare il termine *lock*, dal momento che «blocco» esprime perfettamente il concetto, anche per chi è esperto.

- inode

  Si tratta di un termine costruito appositamente, anche se dalla fusione di termini inglesi. In particolare è difficile stabilire con certezza il significato della lettera «i» iniziale, probabilmente sta per *index*; comunque la diffusione del termine inode è tale per cui non avrebbe senso scomporlo e trasformarlo altrimenti. Per questo non è utile tentare di tradurlo, tanto più che si tratta di un nome costruito ad arte per rappresentare la caratteristica fondamentale dei file system Unix.

- magic number

  Il magic number, come descritto da *magic(4)*, è una realtà presente da molto tempo. Il concetto si avvicina a quello dell'impronta virale utilizzata dai programmi anti-virus, cosa che potrebbe essere descritta come una stringa di riconoscimento. Tuttavia, qualunque traduzione ne cancellerebbe la storia.

- memoria cache

  Memoria cache si usa generalmente così in italiano e non si può tradurre come «memoria tampone» che invece si riferisce al concetto di *buffer*. È da notare che cache viene dal francese. La traduzione «memoria di transito» può servire eventualmente come spiegazione, dal momento che rende abbastanza il concetto.

- news (Usenet)

  Questo termine è intraducibile e si riferisce al servizio offerto dalla rete Usenet: quello di distribuire le news. In questo senso, piuttosto che parlare di «servizio Usenet», è meglio riferirsi a un «servizio di gestione delle news».

- pipe, pipeline

  Di per sé si tratta di «condotti» e «condutture»... tuttavia, forse è meglio non tradurli.

- ping

  Il ping è inteso come l'azione di inviare una richiesta di eco a un nodo di rete, utilizzando il protocollo ICMP. In pratica, si fa il ping attraverso il comando '**`ping`**'. Dal momento che si tratta di un abbinamento con il ping-pong, sarebbe inopportuna la traduzione, a meno di volere essere più chiari, nel qual caso si può parlare di «richiesta di eco».

- pixel

  Dipende dal contesto: se il momento è discorsivo, si può tradurre come «punto grafico», tanto più che la dimensione di un punto del genere non è stabilita, ma dipende dalle caratteristiche del mezzo di visualizzazione.

- proxy

  Il proxy sarebbe il «procuratore» o il «procacciatore» di qualcosa. In italiano è improponibile l'uso di questo genere di traduzioni per indicare il concetto riferito ai servizi di un demone in un sistema operativo.

  Tuttavia, alle volte questo termine è utilizzato in situazioni che non sono particolarmente specifiche; in questi casi si potrebbe parlare di «intermediazione» e di «intermediario».

- record

  Questo termine viene usato spesso nel documento per indicare delle «righe» di file strutturate in campi, che contengono un'informazione completa su qualcosa.

- script

  Lo script, inteso come un programma scritto in un file di testo che viene eseguito per opera di un interprete, è un termine che non ha un equivalente in italiano nell'uso corrente. Inoltre, c'è da considerare che non ci sono difficoltà particolari nell'inserimento in una frase in italiano; anche la pronuncia non è difficile.

- stack

  Il termine stack viene usato spesso per fare riferimento precisamente a quella parte di memoria utilizzata per salvare i registri del microprocessore nell'immagine dell'eseguibile, mentre questo è in funzione. Per rendere chiaro il concetto, conviene parlare di «stack del processo»; negli altri casi dovrebbe essere meglio utilizzare l'espressione «pila».

- standard input, standard output, standard error

  Si tratta di termini praticamente già tradotti, dove eventualmente si dovrebbero solo invertire le parole (input standard, output standard, ecc.). Il problema sta nella traduzione di standard error, che in questo modo diventerebbe «errore standard». Una forma del genere potrebbe far pensare all'«errore che fanno tutti», perché è «standard». Forse si potrebbe risolvere aggiungendo un trattino, ma poi occorrerebbe farlo anche per gli altri. Per il momento, questi termini sono lasciati così come sono in questo documento, senza tentare alcuna traduzione.

- task

  Probabilmente, l'uso del termine task è inevitabile, a meno di grosse arbitrarietà linguistiche. Tra le altre cose, task ha il vantaggio di essere breve e facile da pronunciare all'interno di un testo italiano.

## 15.2 Glossario

Nelle sezioni seguenti sono annotati alcuni termini tecnici, nella maggior parte dei casi si tratta di termini in lingua inglese a cui si affiancano le loro traduzioni o traslazioni possibili in italiano, assieme a qualche commento. Le sezioni servono a distinguere i contesti.

L'asterisco che appare a fianco di alcune definizioni, serve a indicare quelle più deboli, o che comunque sono delimitate nel sorgente all'interno di elementi del tipo:

```
<special special="ttid">>termine</special>
```

In questo modo sono più facili da tenere sotto controllo quando si stampa una bozza, senza lasciare tracce nella composizione finale standard.

### 15.2.1 Unità temporali

Le definizioni legate al conteggio del tempo rappresentano un concetto molto importante, specialmente per gli astronomi. In questo settore si sono sviluppati una serie di acronimi in lingua inglese, che a volte vengono anche tradotti in italiano. In generale, non è opportuno utilizzare acronimi tradotti, che comunque esistono.

- UT, universal time ---> tempo universale

  È il tempo misurato con metodi astronomici, corrispondente al tempo solare medio del meridiano zero (quello passante per l'osservatorio astronomico di Greenwich)

- UTC, universal time coordinated ---> tempo universale coordinato
- CET ---> tempo medio dell'europa centrale
- CEST

  È l'ora estiva in anticipo di un'ora sul tempo CET.
- MET ---> CET

  MET è la vecchia sigla che è stata sostituita da CET.
- time zone ---> fuso orario

  zone ---> fuso
- daylight saving time ---> ora estiva

  È di uso comune chiamare «ora legale» l'orario anticipato di un'ora rispetto al tempo solare che si adotta dalla primavera all'autunno; tuttavia, sarebbe più corretto chiamarlo «ora estiva», chiamando corrispondentemente «ora invernale»[1] l'ora nel resto dell'anno, perché entrambe queste ore sono adottate per legge con tutti gli effetti civili, legali, ecc., quindi sono entrambe ore «legali». Perciò l'aggettivo «legale» non le differenzia.
- timestamp - -> informazione data-orario

  Il *timestamp* è il timbro contenente la data e l'ora dell'istante in cui questo timbro è stato fatto. La traduzione indicata rappresenta un modo imperfetto per esprimere il concetto. Il termine «datario» non è appropriato, dal momento che si riferisce allo strumento per timbrare e non al timbro che si ottiene; inoltre, serve a rappresentare una data, senza l'informazione oraria che invece è determinante nel termine inglese.

  Pare che nell'ambiente militare si usi la forma «gruppo data-orario».

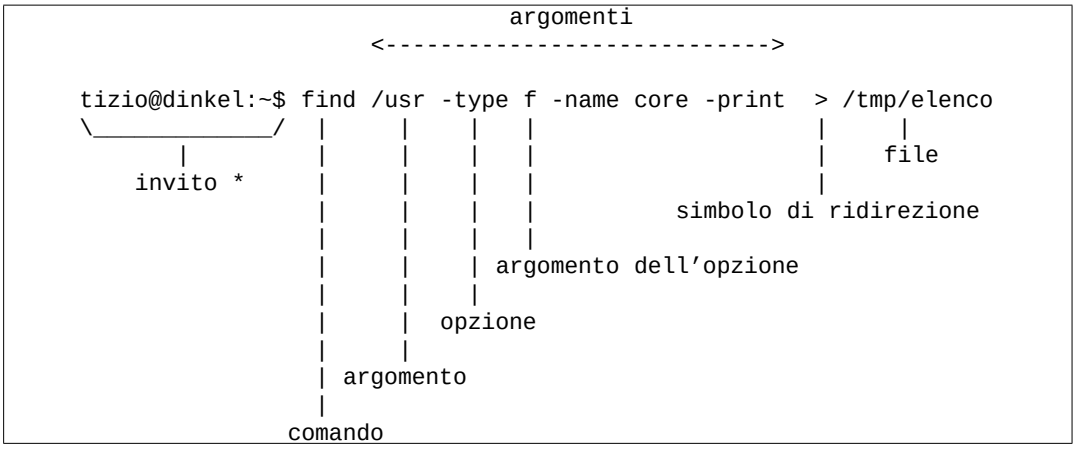
Vedere anche: *Il Tempo di Internet* di Fabrizio Pollastri <*http://toi.iriti.cnr.it/it/toi.html*> e il glossario relativo <*http://toi.iriti.cnr.it/it/glossary.html*>.

### 15.2.2 Comandi e processi elaborativi

- riga di comando

  La riga di comando è quella riga che segue l'invito di una shell. La figura 15.1 raccoglie le definizioni riferite alle varie parti di questa riga.

  Figura 15.1. Descrizione delle varie parti di un comando.




[1] Anche la definizione «ora solare» è imprecisa, perché l'ora solare vera e propria non è la stessa su tutto il fuso orario a cui viene invece applicata

- prompt ---> invito *

  In passato è stata usata la definizione «segnale di pronto» e anche «invito»; questa ultima forma ha il pregio di essere una buona traduzione del significato che ha *prompt*, anche se ha il difetto di non essere utilizzata in generale.

- utility ---> programma di utilità *, programmi di utilità * ---> utilità *

  utility ---> programma di servizio *

  In inglese si utilizza l'espressione «utility» per fare riferimento alla fornitura di servizi fondamentali come l'acqua, l'elettricità, il gas. In questo senso, dovrebbe essere più appropriata la traduzione programma di servizio, piuttosto di parlare di «utilità» come si è sempre fatto (non sapendo di cosa si trattava).

  Resta comunque necessario tenere presente che questa definizione non si può abbreviare semplicemente con «servizio», perché questo porterebbe a fare confusione con i servizi offerti da demoni, attraverso un socket di dominio Unix o una porta di rete.

- pipe, pipeline

  Vedere 15.1.1.

- foreground (process) ---> (processo elaborativo) in primo piano *

  Dal momento che l'uso in questa forma non è molto diffusa, anche se è abbastanza intuitiva, può essere opportuno indicare tra parentesi il termine originale in inglese almeno la prima volta.

- background (process) ---> (processo elaborativo) sullo sfondo *

  Purtroppo, questa forma non è comprensibile immediatamente, per cui si può rendere necessario riproporre tra parentesi il termine originale in inglese almeno la prima volta, o comunque quando il contesto lo richiede per chiarezza.

- task

  Vedere 15.1.1.

- multitasking ---> multiprogrammazione * ---> in multiprogrammazione * ---> multiprogrammato *

  Si tratta di un termine italiano di tipo accademico; probabilmente potrebbero andare bene forme del tipo «sistema che opera in multiprogrammazione» o semplicemente «sistema in multiprogrammazione», per tradurre il concetto di «sistema *multitasking*».

- singletasking --->monoprogrammazione * ---> in monoprogrammazione * ---> monoprogrammato *

  Si riferisce a un sistema operativo che non funziona in multiprogrammazione.

- applicazione concorrente *

  Un programma che genera processi differenti gestiti simultaneamente (pseudo-simultaneamente).

  – applicazione multithread

    Un programma che si scinde in processi distinti, che però funzionano nello stesso contesto di dati. I processi generati sono i *thread* a cui si fa riferimento.

  – applicazione parallela

    Un programma che si scinde in processi distinti, funzionanti in contesti indipendenti, comunicanti tra di loro attraverso dei messaggi.

  - applicazione distribuita

    Un programma che si scinde in processi distinti, eseguiti da macchine diverse, connesse in rete e comunicanti attraverso un protocollo appropriato.

- linguaggio concorrente *; linguaggio di programmazione concorrente *

  Il linguaggio di programmazione che consente la programmazione concorrente con appositi costrutti.

- programmazione concorrente *

  Programmazione di applicazioni concorrenti.

- multielaborazione *

  L'azione di un sistema composto da più CPU che lavorano assieme nello stesso elaboratore, oppure su elaboratori distinti connessi in rete.

- programma sequenziale

  Un programma che corrisponde a un processo singolo.

- runlevel ---> livello, livello di esecuzione *

- exit status ---> valore di uscita *

- boot ---> avvio, caricamento (del sistema operativo)

- Init ---> procedura di inizializzazione del sistema *

  La definizione riguarda il sistema che controlla sia l'avvio che l'arresto del sistema.

  - procedura di avvio del sistema *

    Questa forma viene usata per distinguere all'interno della procedura di inizializzazione del sistema la sequenza delle operazioni nel momento dell'avvio del sistema operativo.
  - procedura di arresto del sistema *

    Questa forma viene usata per distinguere all'interno della procedura di inizializzazione del sistema la sequenza delle operazioni nel momento dell'arresto del sistema operativo.

- Init ---> processo iniziale

  Quando il contesto si riferisce al processo numero uno.

- shutdown ---> arresto del sistema

- spool ---> coda

  La traduzione non è perfetta, ma rappresenta il concetto.

- print job ---> processo di stampa *

- log ---> registro, registro elettronico ---> registrazione degli eventi *

  - to log ---> registrare
  - system log ---> registro del sistema *
  - log file ---> file delle registrazioni *, file di registrazioni *, file per le registrazioni *
  - log archive ---> archivio delle registrazioni *

  È da osservare che la forma «registro elettronico» viene usata frequentemente nei contratti e nei documenti formali.

- interrupt ---> interruzione

  In generale, la prima volta è meglio mettere tra parentesi il termine originale inglese.

- front-end - -> parte frontale *, - -> programma frontale *

  back-end - -> parte terminale *, - -> programma terminale *

  La traduzione non è perfetta, dal momento che *front-end* e *back-end* rappresentano un concetto. In certe situazioni, il *back-end* può essere costituito da un gruppo di programmi, come nel caso delle copie di '`postgres`' avviate da '`postmaster`'. In questi casi, volendo continuare a parlare di programma terminale, occorrerebbe utilizzare il plurale.

  In certe situazioni, *front-end* viene usato in modo improprio anche in inglese; in quei casi, non ha senso la traduzione proposta qui.

### 15.2.3 Memoria centrale e virtuale

- cache memory ---> memoria cache *

  Vedere 15.1.1.

- buffer ---> memoria tampone *

  La traduzione di *buffer* con «tampone» è interdisciplinare. Il termine *buffer*, tradotto con «tampone», si usa persino in chimica e biologia, rappresentando un concetto simile. Tuttavia, è meglio se quando si scrive si pensa che chi legge non sia necessariamente al corrente di questa ambivalenza, per cui conviene ricordare tra parentesi il termine inglese.

- swap ---> scambio *

  Il contesto deve servire a comprendere il significato della parola «scambio». Per esempio: scambio della memoria, area di scambio (della memoria), partizione di scambio (della memoria) file di scambio (della memoria),...

- nvram ---> memoria non volatile

### 15.2.4 Hardware

- computer ---> elaboratore, sistema di elaborazione - -> sistema

- slot ---> alloggiamento

  Il termine *slot* può avere diverse traduzioni a seconda del contesto, pur restando nell'ambito dell'hardware. Per esempio, potrebbe essere espresso come «connettore» e anche «zoccolo», se si intende fare riferimento proprio al sistema di contatti e non anche allo spazio e alle guide delle schede che vi vengono inserite.

- controller ---> unità di controllo *, scheda di controllo *

  L'unità di controllo può essere una scheda o essere una parte integrata nella scheda madre. Al contrario, la scheda di controllo precisa che si tratta di una scheda distinta.

- terminale a caratteri, terminali a caratteri

- adapter, driver (inteso come unità hardware) ---> adattatore

  Questo è il caso di un'interfaccia hardware di qualche tipo, specialmente quando si tratta di una scheda. Si potrebbe parlare di «adattatore SCSI», «adattatore grafico»,...

  - scheda SCSI, interfaccia SCSI ---> adattatore SCSI
  - scheda video, scheda grafica ---> adattatore grafico

### 15.2.5 Dispositivi

In generale, si può distinguere tra dispositivo fisico e un dispositivo logico, per indicare rispettivamente l'hardware di un componente e il file di dispositivo relativo, che rappresenta la visione virtuale offerta dal kernel.

- device ---> dispositivo

  Distinguendo eventualmente in «fisico» o «logico», come accennato.

- device file ---> file di dispositivo
- device driver ---> gestore di dispositivo
- major number ---> numero primario
- minor number ---> numero secondario
- device number ---> numero di dispositivo
- driver ---> gestione di..., gestore *

  In generale, se possibile è meglio parafrasare in modo da essere chiari sul significato della «gestione» a cui si fa riferimento. Si deve tenere presente che in alcune circostanze potrebbe non essere conveniente la traduzione.

- to drive ---> gestire

### 15.2.6 Codifica

- tab ---> carattere di tabulazione
- newline ---> codice di interruzione di riga *

  Questa forma così prolissa serve a indicare il codice necessario a terminare una riga di un file di testo normale, in base alle esigenze del sistema operativo o comunque secondo il contesto. Ciò senza usare il termine *newline*, che a volte alcuni autori di lingua inglese utilizzano per identificare precisamente il codice <*LF*>, indipendentemente da qualunque circostanza.

- escape

  Non conviene tentare di tradurre il termine escape, soprattutto per la sua ambiguità, che lo fa utilizzare in tante situazioni. Vale la pena di annotare alcune forme tipiche in cui può essere utilizzato in italiano.

  - codice di escape

    Quando si tratta di una sequenza di escape che rappresenta qualcosa che esprime un codice speciale, come quello che non ha una corrispondenza simbolica (non è stampabile).

  - sequenza di escape

    Rappresenta qualcosa che si esprime con un carattere di «escape» iniziale, seguito da qualcosa d'altro. In generale, viene usata questa espressione in tutti i casi esclusi quelli in cui la sequenza di escape serve a rappresentare un codice particolare.

- eof, EOF ---> codice di EOF

  EOF è un codice che di solito corrisponde a *<EOT>*, ma in generale dipende dalla piattaforma, più o meno come accade per il codice di interruzione di riga.

### 15.2.7 Tastiera

La tabella 15.3 raccoglie i nomi che sembrano più appropriati per i tasti delle tastiere comuni.

Tabella 15.3. Elenco dei nomi di alcuni tasti.

| Originale inglese | Definizioni possibili in italiano |
|---|---|
| Esc, Escape | Esc |
| Return | Invio |
| Ctrl, Control | Ctrl, Controllo |
| Meta | Meta |
| Alt | Alt |
| Alt Gr | AltGr, Alt Gr |
| Shift | Maiuscole |
| Caps-lock | Fissa-maiuscole |
| Compose | Comp, Composizione |
| PgUp | Pagina su |
| PgDn | Pagina giù |
| Home | Inizio |
| End | Fine |
| Ins, Insert | Ins, Inserimento |
| Del, Delete | Canc, Cancellazione |
| Num Lock | BlocNum |
| Scroll Lock | BlocScorr |
| Print Screen | Stampa |
| F1, F2,... | F1, F2,... tasti funzione, tasti funzionali |
| Tab | Tab, Tabulazione -- per la dattilografia è «tabulatore» |
| Space | Barra spaziatrice, barra spazio, spazio |

Le combinazioni di tasti vengono rappresentate usando il segno '**+**' per indicare una combinazione, mentre le sequenze di tasti vengono semplicemente elencate. Per esempio, [ *Ctrl+x* ][ *Ctrl+y* ] rappresenta la combinazione del tasto di controllo con la lettera «x», quindi il rilascio dei tasti e la combinazione successiva del tasto di controllo e della lettera «y». In presenza si combinazioni particolari, è bene spiegare tra parentesi ciò che si intende. Quando le combinazioni includono delle lettere alfabetiche, se non conta il fatto che siano maiuscole o minuscole, si rappresentano usando l'alfabeto minuscolo.

- key binding ---> associazione dei tasti *

  Il significato attribuito a tasti particolari o a combinazioni di questi.

- interrupt character ---> carattere interrupt

  Per comprenderne il senso, si può consultare la pagina di manuale *stty(1)*.

### 15.2.8 File di testo

- patch (file) ---> file di differenze *

  Trattando di *patch* si può parlare anche di «modifiche», «variazioni», «aggiornamenti» e simili, in base al contesto. Tuttavia, viene usata prevalentemente la definizione «file di differenze» come sostituto di «file di *patch*».

  Quando si «applicano», si fa riferimento prevalentemente a «modifiche», senza richiamare nuovamente il termine «differenze».

- regular expression ---> espressione regolare *

- '`/etc/motd`' ---> file contenente il messaggio del giorno

- '`/etc/issue`' - -> file contenente il messaggio di pubblicazione

  Sembra che il file '`/etc/issue`' servisse per fare apparire l'informazione sul nome e il numero di versione del sistema operativo. In questo senso, si potrebbe parlare di «numero di edizione», o di «pubblicazione», come se si trattasse di una rivista.

### 15.2.9 Archiviazione e pacchetti applicativi

- archive (file) ---> archivio ---> archivio compresso

  Si fa riferimento a un file utilizzato per archiviare file e directory, come quello generato da '**`tar`**'. Un «archivio» è un file del genere realizzato in qualunque forma, anche compresso, mentre un «archivio compresso» è precisamente un file che ha subito una forma di riduzione (senza perdita).

  Sono archivi anche i file dei pacchetti di applicazioni delle varie distribuzioni GNU/Linux: archivi Slackware, archivi RPM, archivi Debian...

  - archiviazione

    L'azione con cui si crea un archivio (compresso o meno che sia).

  - estrazione (del contenuto)

    L'azione con cui si estraggono i dati contenuti in un archivio (file, directory e altri oggetti, assieme ai loro attributi).

- package ---> pacchetto (applicativo)

  In questo contesto, il «pacchetto» è ciò che è contenuto in un archivio di una distribuzione GNU/Linux. Per esempio, si può parlare di *archivio* '`bash_2.01.1-4.1.deb`' e di *pacchetto* '**`bash`**' (oppure Bash, se si vuole essere un po' meno precisi).

### 15.2.10 Dati

- magic number

  Vedere 15.1.1.

- record

  Vedere 15.1.1.

- standard input, standard output, standard error

  Vedere 15.1.1.

- database ---> base di dati *, basi di dati *

  In italiano si utilizza prevalentemente quando si tratta veramente di *database*, ovvero di *relazioni*. In italiano è frequente anche l'uso della forma «base dati», togliendo il «di».

- database ---> elenco, registro, tabella

  Quando il termine *database* viene usato in modo improprio, potrebbe essere corretto l'uso di altri termini in funzione del contesto.

- data type ---> tipo di dati, tipi di dati

- checksum - -> codice di controllo *

  Il *checksum* indica letteralmente una «somma di controllo», solo che nel tempo si è esteso il suo significato includendo anche altre forme di controllo basate su operazioni di tipo diverso. A seconda delle circostanze si possono distinguere traduzioni differenti, che servono a precisare il tipo di controllo che viene attuato attraverso il *checksum*.

  – codice di controllo *

    Questa è probabilmente la traduzione migliore che potrebbe adattarsi alla maggior parte delle circostanze, dal momento che non viene specificato il modo in cui si ottiene il valore di controllo, non si stabilisce nemmeno la sua forma (numerica, alfabetica, ecc.); inoltre, non si stabilisce la sua dimensione.

  – carattere di controllo, cifra di controllo *

    In tal caso il valore utilizzato per il controllo è rappresentato da un solo carattere, oppure precisamente da una cifra numerica.

  – somma di controllo *

    Questa è la traduzione letterale del significato di *checksum*, però il suo uso dovrebbe essere riservato al caso in cui la funzione che genera il codice di controllo è basato su un procedimento di somme.

  – campo di controllo *

    Quando l'informazione che funge da controllo è contenuta in un «campo».

  – controllo

    Quando il contesto si riferisce all'azione di verificare qualcosa in base a un codice di controllo, ci si può limitare a usare il termine «controllo».

- MD5 digest, MD5 message digest - -> firma MD5

  In un certo senso, un *MD5 digest* è un riassunto matematico di un messaggio, giustificando il motivo dell'utilizzo del termine *digest*. Oltre a questo, la stessa sigla «MD» sta per *Message digest*.

- upload, download ---> carico, scarico

  I termini inglesi *upload* e *download* dovrebbero derivare dalle operazioni di carico e scarico delle merci dai mezzi di trasporto.

- octet ---> ottetto

- empty string ---> stringa nulla

- stringa vuota ---> stringa nulla

  Per coerenza, è bene usare una sola definizione.

- trigger ---> grilletto *

### 15.2.11 Crittografia e firma elettronica

- in chiaro

  cifrato, in cifra

  Nel primo caso si fa riferimento a un'informazione che si presenta nella sua condizione normale, per la sua leggibilità o per l'accessibilità del suo contenuto; nel secondo caso, si tratta di un'informazione cifrata.

- cipher ---> cifratura

  encrypted ---> cifrato

  encryption ---> cifratura

  La traduzione esatta di *encryption* è crittografia, che però è un sinonimo di cifratura. L'intenzione è quella di utilizzare in modo univoco questo tipo di tecnica.

- crittografia

  Si preferisce riservare questo termine per fare riferimento al concetto generale, che si concretizza nell'uso della cifratura dei dati.

- decrittazione

  Dovrebbe essere l'operazione attraverso cui si riesce a decifrare un'informazione senza conoscerne la chiave o il cifrario.

- Distinguishing Name, DN ---> nome distintivo *

  Certificati X.509.

- Common Name, CN ---> nome comune *

  Certificati X.509, campo CN del nome distintivo.

### 15.2.12 Linguaggi di programmazione e compilatori

I nomi attribuiti ai tipi di dati di ogni specifico linguaggio di programmazione, non possono essere tradotti, perché si tratta di parole chiave. Tuttavia, in un ambito discorsivo, ha senso utilizzare delle definizioni comprensibili. La tabella 15.4 mostra un elenco di quelle più comuni.

Tabella 15.4. Elenco delle definizioni possibili riferite ai tipi di dati più comuni.

| | |
|---|---|
| char | carattere |
| int | intero |
| float | a virgola mobile (singola precisione) |
| double | a virgola mobile e doppia precisione |

I nomi delle strutture di controllo del flusso e delle altre istruzioni che condizionano il flusso delle istruzioni, possono essere tradotti in alcuni casi, riferendosi al comportamento delle istruzioni a cui si fa riferimento. La tabella 15.5 riassume queste possibilità.

Tabella 15.5. Elenco delle definizioni e dei nomi riferiti alle strutture di controllo del flusso delle istruzioni.

| | |
|---|---|
| go to | salto incondizionato |
| if | condizione, struttura condizionale |
| switch, case | selezione |
| while | iterazione, ciclo iterativo (condizione iniziale) |
| until | iterazione, ciclo iterativo (condizione finale) |
| for | iterazione enumerativa, ciclo enumerativo |
| break | salto, interruzione |

La figura 15.2 raccoglie le definizioni riferite alla dichiarazione delle funzioni nei linguaggi di programmazione; la figura 15.3 fa riferimento alle definizioni utili nella chiamata di una funzione.

Figura 15.2. Linguaggi di programmazione: dichiarazione delle funzioni.

```
C               int potenza (int x, int y)
                (a)   (b)    (c)   (c)

Pascal          function potenza( x : integer; y : integer ) : integer;
                          (b)      (c)          (c)            (a)

Scheme          (define (potenza x y) ... )
                          (b)   (c)
(a) tipo restituito
(b) nome della funzione
(c) parametri formali
```

Figura 15.3. Linguaggi di programmazione: chiamata delle funzioni.

```
                C               z = moltiplica (x, y);
                                 (a)   (b)       (c)

                Pascal          z := moltiplica( x, y );
                                   (a)  (b)        (c)

                Scheme          (set! z (moltiplica  x y))
                                             (b)      (c)
                (a) assegnamento
                (b) funzione
                (c) parametri
```

- assegnamento

  Per indicare il fatto che si assegna un valore a una variabile, si pone l'alternativa di usare «assegnazione» o «assegnamento». Si è scelta questa seconda alternativa.

- parametro formale, parametro

  Nella dichiarazione di una funzione (o di una procedura), l'indicazione delle variabili di scambio, assieme alle informazioni sulle loro caratteristiche, viene indicata come la definizione dei ***parametri formali***.

  Quando si chiama una funzione, gli «argomenti» della chiamata, sono i ***parametri*** della funzione.

- array

  Vedere 15.1.1.

- associative array ---> array associativo

- script

  Vedere 15.1.1.

- script language, scripting language ---> linguaggio script, linguaggio di script

- stream ---> flusso *

  In questo caso, si fa riferimento allo *stream* che rappresenta un file aperto in C. Si distingue tra file aperto e file vero e proprio per il fatto che uno stesso file può essere stato aperto più volte all'interno di un programma.

- filehandle, file handle ---> flusso di file * - -> flusso *

  In questo caso, si fa riferimento a ciò che rappresenta un file aperto in Perl. Valgono le stesse considerazioni fatte per il caso dello *stream*, in C.

- makefile ---> file-make *

  Questa definizione ha il vantaggio di essere comprensibile anche per chi utilizza abitualmente la definizione originale: *makefile*.

- to port ---> adattare

  porting ---> adattamento

  Con questo termine si fa riferimento al lavoro necessario per adattare un programma a un'altra piattaforma rispetto a quella di partenza.

### 15.2.13 Memoria di massa

- directory

  Vedere 15.1.1.

- inode

  Vedere 15.1.1.

- link ---> collegamento *

  - symbolic link ---> collegamento simbolico *
  - hard link ---> collegamento fisico *

- umask ---> maschera dei permessi *

  La documentazione della shell Bash fa riferimento al comando '**`umask`**' come a quello che imposta la «maschera di creazione dei file» per i processi elaborativi. Tuttavia, utilizzando questa definizione si perde di vista il compito preciso di questa maschera: quello di eliminare alcuni permessi in modo predefinito.

- sticky (bit) ---> (bit) Sticky

  In pratica, viene usato sempre con l'iniziale maiuscola in modo da abbinarlo facilmente agli altri «s-bit»: SUID, SGID e Sticky.

  Quando *sticky* viene usato in altri contesti, si potrebbe tradurre come «adesivo».

- mode ---> modalità dei permessi

  Evidentemente si fa riferimento ai 12 bit che definiscono i permessi di un file, lasciando da parte la proprietà dei file.

- permessi di accesso

  Si tratta degli ultimi nove bit della modalità dei permessi, in cui si regolano proprio gli accessi a file e directory.

- mount, unmount ---> dipende dal contesto
  - mount ---> montaggio * - -> innesto * - -> inserimento * - -> collegamento *
  - unmount ---> smontaggio * - -> distacco *
  - mount point ---> punto di innesto *
  - directory di innesto *
  - to mount ---> montare
  - to unmount ---> smontare
- home directory

  La traduzione di questa definizione non è possibile in un modo unico, dal momento che si possono presentare situazioni differenti:
  - ---> directory personale *

    quando si tratta di un utente umano, oppure quando si dà una personalità virtuale all'utente fittizio;
  - ---> directory iniziale *

    quando si tratta di un utente fittizio riferito a un servizio, specialmente se questa directory è effettivamente l'«inizio» della gerarchia dell'applicativo (è evidente che questa definizione può essere usata solo se il contesto è compatibile).
- root ---> dipende dal contesto
  - root directory ---> directory radice *
  - root file system ---> file system principale
  - root partition ---> partizione principale *
- path, pathname ---> percorso

  I termini *path* e *pathname*, quando riguardano il percorso di un file o di una directory, hanno una differenza sottile che non sempre viene tenuta in considerazione nel modo corretto: il *pathname* dovrebbe essere un percorso che contiene l'informazione dell'oggetto finale (il file o la directory finale che si vuole indicare); il *path* dovrebbe essere il percorso della directory che contiene un oggetto a cui si fa riferimento.

  A seconda dell'opportunità o meno, si può usare anche la forma «nome di percorso».
- percorso relativo

  percorso assoluto

  I due casi fanno riferimento rispettivamente a un percorso che parte dalla posizione di partenza e un percorso che parte invariabilmente dalla radice. In generale, la forma «percorso completo» è ambigua, perché può far pensare al *pathname*, pertanto è meglio evitarla.
- ramdisk, RAM disk ---> disco RAM *
- backup ---> dipende dal contesto

  La parola *backup* è il classico esempio di termine conciso e ambiguo della lingua inglese. Per tradurlo occorre utilizzare definizioni differenti a seconda del contesto. Segue un elenco di definizioni che potrebbero essere utilizzate a seconda del contesto particolare e a seconda del gusto del momento.

  - copia di sicurezza, salvataggio

    In questo caso si intende il *backup* come la copia che si fa per premunirsi contro le perdite di dati accidentali.

  - copia di sicurezza di versioni precedenti

    Alcuni programmi che copiano o spostano dei file, se incontrano altri file con lo stesso nome nella destinazione, cambiano il nome di questi ultimi, aggiungendo un'estensione simbolica (di solito una tilde, o il simbolo '`#`'). Queste sono delle copie di *backup*, nel senso che sono le copie di sicurezza delle versioni precedenti di quei file.

  - copia di riserva

    La copia di riserva è una copia che si affianca all'«oggetto» che si utilizza (il file, il dischetto, ecc.), nel caso questo risulti danneggiato.

- Linux native (partition) ---> (partizione) Linux-nativa *
- Linux swap (partition) ---> (partizione) Linux-swap *

### 15.2.14 Utenza

- user ---> utente, utilizzatore

  Vale la pena di distinguere tra l'utente inteso come entità che accede al sistema, rispetto all'utilizzatore (umano) di qualcosa.

- utente comune

  L'utente comune dovrebbe essere inteso come l'utente di un sistema Unix che non ha privilegi particolari, ovvero un utente che non è l'amministratore (né '`root`', né un altro amministratore di qualche parte particolare del sistema).

- utilizzatore normale

  L'utilizzatore normale dovrebbe essere quella persona che utilizza un accesso o un servizio senza grandi pretese e senza competenze speciali.

- utente normale

  In alcuni casi, la definizione «utente comune» non va bene, per esempio quando si parla degli utenti normali del servizio WU-FTP.

- user name ---> nominativo-utente

  Si tratta del nome che un utente utilizza per identificarsi e accedere al sistema. Al nominativo-utente si abbina una parola d'ordine.

- account ---> dipende dal contesto

  Il termine *account* non è traducibile in un modo solo per tutti i contesti in cui si può usare in inglese. Segue un elenco di definizioni che potrebbero essere utilizzate a seconda del contesto particolare e a seconda del gusto del momento.

  - utente -- quando si fa riferimento a un «utente logico» del sistema;
  - utente registrato (nel sistema);
  - utenza -- quando si vede l'aspetto contabile della faccenda, ovvero quando l'*account* è più vicino all'idea di un contratto per ottenere l'accesso;
  - accesso;

  - recapito -- nella posta elettronica;
  - profilo (personale) -- quando si fa riferimento a un file di configurazione collocato nella directory personale;
  - privilegi (di un certo utente) -- quando l'utente serve a fare o a evitare che sia fatto qualcosa di particolare;
  - identità (di un utente).

- client, server ---> cliente *, servente *

  I termini cliente e servente sono ambigui, sia in italiano che nell'originale inglese. Il problema nasce dal fatto che dipende dal contesto cosa sia «cliente» e cosa sia «servente». In un testo scritto in lingua italiana, dovrebbe essere auspicabile il chiarimento del contesto, come viene proposto nell'elenco seguente:

  - programma cliente, programma servente

    quando si fa riferimento a un programma che utilizza o che fornisce un servizio di qualche tipo;
  - nodo cliente, nodo servente

    quando si fa riferimento a una connessione in cui si distingue tra nodi che chiedono un servizio e nodi che forniscono un servizio, tenendo presente che all'interno dei nodi ci sono ovviamente dei programmi cliente e dei programmi servente;
  - elaboratore cliente, elaboratore servente

    quando si fa riferimento all'elaboratore in cui si utilizza un programma cliente o un programma servente, senza voler porre un'enfasi particolare sul collegamento di rete.

### 15.2.15 Documentazione

- man page ---> pagina di manuale *

  Lo Unix AT&T aveva un manuale cartaceo, diviso in sezioni, dove ogni comando costituiva una sottosezione. La composizione del manuale avveniva attraverso Troff ed era disponibile anche tramite il comando '**`man`**', abbreviazione di *manual*.

- on-line help ---> guida interna

  Si può considerare anche la possibilità di usare la forma «guida in linea», se appropriato.

- help ---> guida, guida interna

### 15.2.16 Interfaccia grafica

- desktop ---> superficie grafica * ---> scrivania grafica *

  A seconda del contesto, può essere più appropriata la definizione di superficie grafica, oppure di scrivania grafica. Per la precisione, la superficie dello schermo, quando viene usato con un gestore di finestre comune, è da intendersi semplicemente una superficie grafica, mentre un sistema più complesso (come Gnome) può essere definito come scrivania grafica.

- session manager ---> gestore di sessione *

  Si tratta per esempio di Gnome o KDE, visti nell'ambito del controllo della sessione di lavoro con il sistema grafico X. Si parla di sessione quando si usa un *display manager*, come Xdm, Gdm, Kdm e simili.

- display manager ---> sistema grafico di autenticazione

  Si tratta per esempio di Xdm, Gdm, Kdm e simili.

- root window ---> finestra principale *

  Utilizzando questa traduzione, occorre fare attenzione a non usare la stessa definizione per fare riferimento alla finestra più importante di un programma che può presentare diversi componenti su più finestre.

- screen saver ---> salva-schermo

- window manager ---> gestore di finestre *

- stazione grafica *

  X utilizza una definizione un po' contradditoria dei componenti di ciò che qui viene chiamato stazione grafica. Con questa definizione si fa riferimento al servizio offerto da un servente X; in tal modo, se ci sono più serventi X in funzione, ci sono altrettante stazioni grafiche virtuali, esattamente come accade per le console virtuali. In generale, X fa riferimento al *display* per indicare la stazione grafica, solo che poi, quando si tratta di indicare anche lo schermo, si utilizza l'opzione o la variabile di ambiente '**`DISPLAY`**', mentre in questo caso sarebbe opportuno parlare di «schermo» (*screen*) in modo preciso.

- pulsante grafico

  Quando si tratta di un tasto virtuale che appare sullo schermo.

- checkbox ---> casella di spunta

- mouse pointer, mouse cursor ---> puntatore del mouse

  Questo sembra essere un modo elegante per specificare che non si tratta del cursore all'interno del testo.[2]

### 15.2.17 Rete e comunicazioni

- datagram - -> datagramma

  Si tratta dei pacchetti di un protocollo non connesso (UDP).

- bridge

  Vedere 15.1.1.

- switch ---> commutatore di pacchetto *

  La traduzione non è diffusa, ma il termine originale è anche troppo generico.

- router

  Vedere 15.1.1.

- gateway

  Vedere 15.1.1.

- proxy

  Vedere 15.1.1.

- route ---> instradamento

[2] Potrebbe essere interessante anche l'idea di «mirino» del mouse.

- to route ---> instradare
- regola di instradamento *

  Una voce nella tabella degli instradamenti.
- Unix domain socket ---> socket di dominio Unix - -> socket di tipo Unix *

  Meglio la prima delle due possibilità.
- to forward ---> inoltrare - -> proseguire

  In generale, «inoltrare» è la traduzione corretta, a parte una situazione particolare: nella posta tradizionale, quando una corrispondenza deve essere inviata a un indirizzo diverso da quello stabilito originariamente, questa «viene proseguita». Infatti, il problema si pone nel momento della consegna della corrispondenza: il postino viene a sapere che il destinatario ha cambiato indirizzo, oppure la stessa persona che l'ha ricevuta la reimbuca dopo aver modificato l'indirizzo di destinazione. Di conseguenza, sarebbe giusto dire che «si prosegue» un messaggio di posta elettronica quando questo, una volta giunto alla sua destinazione prevista, viene rinviato a un'altra destinazione.
- relay ---> relè *
- link (HTML) ---> riferimento, riferimento ipertestuale *, collegamento ipertestuale *

  In generale, i due termini, riferimento ipertestuale e collegamento ipertestuale, sono la stessa cosa. Eventualmente, a collegamento ipertestuale si può dare un'enfasi locale, mentre a riferimento ipertestuale un significato più lontano. In pratica, un riferimento interno a una stessa pagina HTML, o ad altre pagine che compongono un insieme ben organizzato, sarebbe un collegamento ipertestuale, mentre un riferimento a una risorsa esterna sarebbe un riferimento ipertestuale. Volendo evitare di fare confusione, conviene usare una definizione sola e precisamente riferimento ipertestuale.
- link (IPv6) ---> collegamento di rete *
- computer host ---> elaboratore host, host - -> nodo di rete *, nodo * - -> stazione

  In questo caso si tratta di un elaboratore connesso in rete che in qualche modo ospita qualche servizio. Nel testo si preferisce usare il termine «nodo di rete» o soltanto nodo.

  Il termine *host*, viene usato in particolare nella documentazione RFC riferita a IPv6 per indicare un nodo che non sia un router. Inoltre, sempre la terminologia riferita a IPv6 indica il nodo come qualunque dispositivo che utilizzi in pratica questo protocollo.

  In italiano si utilizza anche il termine «stazione», seguito da un aggettivo che ne specifica il comportamento. Per esempio, nel capitolo dedicato alla realizzazione di elaboratori senza disco, si parla di stazioni senza disco.
- nodo di rete *, nodo *

  Quando si fa riferimento a un indirizzo nella rete, senza specificare il ruolo che ha ciò che vi corrisponde.
- diskless ---> senza disco

  Si fa riferimento a nodi di rete composti da elaboratori senza un disco locale da cui possa essere montato il file system principale (la directory radice). Questi utilizzano il protocollo NFS per il montaggio di tutto il loro file system.
- netmask ---> maschera di rete (IPv4) *

  Non vengono segnalate le abbreviazioni contenenti solo la parola «maschera».

- IP masquerading ---> mascheramento IP *

  La scelta di utilizzare il termine «mascheramento» come traduzione di *masquerading* in riferimento ai pacchetti IP, è discutibile. In generale, da un punto di vista logico, la traduzione corretta di questo termine dovrebbe essere «travestimento», o anche «camuffamento», dal momento che lo scopo del *masquerading* non è quello di nascondere i pacchetti, ma di farli sembrare appartenenti a un'origine differente. In questo documento si preferisce l'uso di «mascheramento», puntando sulla somiglianza letterale del termine con quello originale inglese, oltre al fatto che comunque si ottiene l'effetto di nascondere i nodi reali da cui hanno origine le comunicazioni.

- name server - -> servizio di risoluzione dei nomi *

  La traduzione fatta in questo modo cambia un po' il contesto: *name server* è un nodo che offre un servizio e non il servizio in sé. Quando si vuole fare riferimento proprio al nodo, si può parlare di servente DNS.

- root domain ---> dominio principale *

  Il dominio di «primo livello» è quello che segue immediatamente quello principale; quindi, il dominio principale si rappresenta con un punto singolo, quando il contesto lo richiede, mentre il dominio di primo livello (che discende da quello principale), noto anche come TLD (*Top level domain*) potrebbe essere: `com`, `edu`, `net`, `org`,...

- packet driver ---> driver di pacchetto

  Si tratta del programma Dos utilizzato per comandare l'interfaccia di rete in modo da offrire ad altri programmi l'accesso alla stessa, attraverso un IRQ software.

- format prefix (IPv6) ---> prefisso di formato *

  Rappresenta l'idea di maschera di rete del sistema IPv6.

- interface identifier (IPv6) ---> identificatore di interfaccia

- group identifier (IPv6) ---> identificatore di gruppo

- mirror ---> sito speculare *, riproduzione speculare *

  Meglio la seconda delle due espressioni.

- mailing-list ---> lista di posta elettronica *, lista

- master ---> principale

  slave ---> secondario

  Questa traduzione va bene quando si tratta di serventi di qualche servizio, in cui uno solo è *master*, mentre tutti gli altri sono *slave*.

- master ---> primario

  slave ---> secondario

  Questa traduzione va bene quando si fa riferimento al servizio DNS, dal momento che in passato, il servente *master* veniva definito *primary*.

- chat script ---> script di chat ---> script di colloquio *

- ISP, provider ---> fornitore di accesso a Internet

  Dal momento che la definizione è estremamente lunga, quando il contesto è chiaro, si potrebbe abbreviare a «fornitore di accesso», o anche solo «fornitore».

- chain ---> punto di controllo *

  Si fa riferimento al firewall Linux, secondo i kernel 2.2.∗ e 2.4.∗, dove questo termine individua un punto di intercettazione dei pacchetti IP, allo scopo di applicarvi delle regole (direttive) che si traducono in obiettivi, ovvero nella sorte dei pacchetti stessi.

- internet superserver, internet service daemon ---> supervisore dei servizi di rete *

  Si tratta praticamente di '**`inetd`**' o di '**`xinetd`**', senza fare riferimento in modo preciso a questo o quel programma.

### 15.2.18 Tipografia

- specie (alfabetica)

  Si tratta di una classificazione dei caratteri in base al tipo di linguaggio per cui sono fatti: latino, cirillico, greco,...

- family - -> famiglia di caratteri - -> stile

  Lo stile è una forma di classificazione estetica di un carattere, contrassegnato da un nome, come per esempio il Times. Il termine «stile» va bene fino a quando si resta all'interno di una stessa specie. Alle volte ci sono delle *font family* che si riferiscono a specie differenti, come il tipo Symbol, o Dingbats. La definizione «famiglia di caratteri» potrebbe andare bene nel caso si voglia mantenere la stessa ambiguità. Questa definizione, famiglia di caratteri, viene anche usata effettivamente, però bisogna ricordare che nel linguaggio tipografico tradizionale italiano, la «famiglia» si riferisce precisamente a un gruppo stilistico con piccole varianti rispetto allo stile a cui appartiene. Bisogna fare attenzione.

- serie, variante seriale

  La serie è la diversificazione formale di uno stesso stile alfabetico. All'interno di uno stile, una serie può essere una variante di forma: il tondo, il corsivo, il neretto,...

- forma

  La forma del carattere: il tondo contrapposto al corsivo, il chiaro contrapposto al neretto e altre varianti (inclinato, chiarissimo, nero, nerissimo, ecc.).

  – pendenza
    Un aspetto della forma del carattere: tondo contrapposto a inclinato.
  – tono
    Un aspetto della forma del carattere: dal chiarissimo al nerissimo.
  – width ---> larghezza
    Un aspetto della forma del carattere: dallo strettissimo al larghissimo.

- body size ---> corpo

  L'altezza del carattere.

- interlinea

  Tecnicamente è la distanza tra le righe che si aggiunge alla distanza minima in funzione del corpo del carattere utilizzato. Tuttavia, con questo termine si fa spesso riferimento alla distanza tra le basi di una riga e della successiva (dattilografia).

- foundry ---> fonderia

- serif ---> grazie, terminali

  In italiano, il termine si usa generalmente al plurale.

- sans serif ---> lineare

  Si tratta di uno stile senza grazie.

- collezione alfabetica

  La distinzione tra maiuscole e minuscole.

- font ---> fonte tipografica, fonte di caratteri ---> fonte ---> tipoplesso

  font ---> carattere ---> tipo di carattere ---> carattere tipografico, carattere da stampa

  Il termine *font* non corrisponde esattamente a qualcosa di ben definito nella tradizione della terminologia tipografica italiana, di conseguenza, la traduzione con il termine «fonte» e i suoi vari abbinamenti è solo una forma di derivazione dall'inglese, altrettanto ambigua. Il termine tipoplesso, sembrerebbe essere il più appropriato, solo che si tratta di qualcosa che risulterebbe incomprensibile ai più.

  La scelta di usare la definizione «tipo di carattere», con tutte le altre varianti, può essere motivata da un contesto non molto impegnato dal punto di vista dei problemi che riguardano la composizione tipografica. In generale, la sua semplicità rende più comprensibile il testo al lettore che non abbia già delle nozioni di tipografia.

- polizza

  L'assortimento completo di caratteri di un corpo determinato. Le polizze compongono il tipoplesso. Nella lingua francese, il termine «police» (polizza) si usa per tradurre il termine inglese *font*.

- scala di corpi

  L'insieme dei corpi in cui può essere reso un certo tipo di carattere.

- traslitterazione

  Traduzione da un alfabeto a un altro, lettera per lettera. Nella traslazione di un testo composto in cirillico traslitterato in carattere latino, l'alfabeto latino è il traslitterante e l'alfabeto cirillico è il traslitterato.

- character set ---> insieme di caratteri *

  Da una discussione era emerso che dovendo scegliere tra «gruppo di caratteri» e «insieme di caratteri» è meglio la seconda forma per vari motivi fondati sulla teoria degli insiemi.[3]

- orientamento della stampa

  In questo modo si può identificare come si stampa su un foglio di carta.

  – portrait ---> verticale
  – landscape ---> orizzontale
  – sea-scape ---> rovesciato
  – up side down ---> sottosopra

- segnatura

  Il numero di fogli che compone un fascicolo nell'ambito di un sistema di rilegatura a filo. In pratica, i fogli stampati vanno piegati a metà e poi cuciti sulla piega, in modo da poter essere sfogliati.

[3] Unicode introduce una terminologia più precisa al riguardo di ciò che un tempo si chiamava *character set*.

### 15.2.19 Unicode

- code point ---> punto di codifica *

  Il simbolo dal punto di vista della codifica.

- code unit ---> unità di codifica *

  L'unità di memoria utilizzata per la rappresentazione della codifica.

- CCS: Coded Character Set ---> insieme di caratteri codificato *

  L'insieme di caratteri codificato attraverso un intero non negativo.

- CEF: Character Encoding Form ---> forma di codifica del carattere *

  Mappa di trasformazione tra l'insieme di caratteri codificato e le sequenze di unità di codifica.

- CES: Character Encoding Scheme ---> schema di codifica del carattere *

  Mappa di trasformazione tra le sequenze di unità di codifica e le sequenze di byte.

- TES: Transfer Encoding Syntax ---> sintassi di codifica per il trasferimento *

  Metodo di trasformazione reversibile di una codifica per il trasferimento dei dati.

- wide char ---> carattere esteso *

- wide string ---> stringa estesa *

### 15.2.20 Grafica

- interleaved ---> interfogliato

- mirror ---> ribaltamento speculare

  Si fa riferimento al ribaltamento dell'immagine che si ottiene come se questa fosse posta davanti a uno specchio.

- offset ---> scostamento, scarto

  L'idea viene dal lavoro di ATO (*Amiga translators' organization*).

- despeckle ---> filtro mediano

- thumbnail ---> provino

  Questa traduzione va bene quando il contesto riguarda la selezione di un'immagine da un elenco di riduzioni, i «provini», come quelli che si fanno in fotografia.

- flood fill ---> campitura

- to flood fill ---> campire

### 15.2.21 Usenet

- feed

  Vedere 15.1.1.

- news

  Vedere 15.1.1.

- newsgroup ---> gruppo di discussione (di Usenet) - -> gruppo

  La definizione «gruppo di discussione» è quella più diffusa, anche se per alcuni potrebbe risultare imprecisa: non sempre si tratta di aree di discussione, potrebbero essere semplicemente dei gruppi per la diffusione di notizie di qualche tipo, senza che si formi una discussione vera e propria.

- news server, discussion host ---> servente di news

  Si tratta di un nodo di rete che offre l'accesso ad alcuni gruppi per mezzo del protocollo NNTP.

- to post ---> spedire (un articolo).

- sito Usenet

  Si tratta di un sito che offre un servizio di accesso alla rete Usenet.

- articolo

  L'articolo è ciò che viene diffuso attraverso Usenet, nei gruppi di discussione verso cui è stato spedito. Non si deve confondere con news, che invece rappresenta il servizio in generale.

### 15.2.22 Localizzazione

- collating sequence ---> sequenza di collazione *

  L'insieme ordinato dei simboli (*collating element*) utilizzati in una localizzazione particolare.

- collating element ---> elemento di collazione *

  Un elemento (un simbolo) di una sequenza di collazione.

- collating symbol ---> simbolo di collazione *

  È il simbolo utilizzato per rappresentare un elemento di collazione nella localizzazione. Di solito si tratta di forme del tipo '`<a>`', '`<b>`', '`<c>`', ecc., come si vede nei file '`/usr/share/i18n/locales/*`'.

- equivalence class ---> classe di equivalenza

  Una classe di equivalenza identifica un gruppo di elementi di collazione (in certi casi si parla di caratteri equivalenti, ma si tratta generalmente di una scorciatoia giustificata solo dal contesto), che devono essere trattati come equivalenti per qualche motivo (di solito ai fini dell'ordinamento). Per esempio, le lettere «e», «è», «é» potrebbero essere trattate come equivalenti.

- character class ---> classe di caratteri

  Una classe di caratteri identifica un insieme dei caratteri attraverso un nome. Si distingue solitamente tra: lettere minuscole, lettere maiuscole, cifre numeriche, caratteri alfanumerici, ecc.

### 15.2.23 Varie

- maintainer ---> curatore
- contributor ---> collaboratore
- implementation ---> realizzazione - -> attuazione, adattamento
- to implement ---> realizzare - -> attuare, adattare
- keyword ---> parola chiave, parole chiave
- retry ---> tentativi ripetuti
- disclaimer ---> liberatoria
- flag ---> opzione (booleana), modalità (booleana), attributo (booleano), variabile (booleana), indicatore

  Purtroppo si possono tradurre in questo modo solo alcune situazioni.
- file manager ---> gestore di file *.

  Si tratta di programmi come Midnight Commander, XFM e simili.
- login ---> accesso, procedura di accesso *
- logout ---> conclusione dell'accesso, conclusione della sessione di lavoro
- screen saver ---> salva-schermo
- hard limit, soft limit ---> limite fisico *, limite logico *
- lock ---> blocco

  Si veda al riguardo la nota sui file di lock nella sezione 15.1.1.
- signal trap ---> cattura di un segnale
- to prepend ---> anteporre

  Si fa riferimento all'aggiunta di qualcosa all'inizio di un flusso di dati, o all'inizio di un file.
- et al ---> et alia ---> e altri - -> e simili, ecc.
- menu ---> menù *

  In generale, su alcuni vocabolari è ammesso l'uso del termine «menu» senza accento. Tuttavia, la norma UNI 6015 (3.2.4), fa espresso riferimento alle «parole polisillabe su cui la posa della voce cade sulla vocale che è alla fine della parola...».

- password ---> parola d'ordine *.

  passphrase ---> parola d'ordine *.

  Diventa difficile trovare una traduzione «perfetta» di questi due termini. Volendo tornare alle origini, la traduzione dovrebbe essere «parola d'ordine». Anche se non è un termine usato, rende l'idea. Evidentemente diventa impossibile tradurre password shadow e altre cose che in realtà fanno riferimento a qualcosa di più complesso (in questo caso, *password* è il nome esteso del file '`/etc/passwd`').

  Nel caso particolare di *passphrase*, diventa impossibile una traduzione secondo il criterio indicato, se non perdendo l'informazione cruciale sulla lunghezza che la parola d'ordine deve avere, non essendo più una sola «parola».

  Va annotato comunque che esiste anche la forma «chiave di identificazione», nota almeno nei vocabolari. Si opta comunque per la traduzione originale anche perché il concetto di identificazione si può confondere con il nome fittizio abbinato a un utente.

- peso - -> massa

  Di solito si confonde il peso con la massa di un corpo. Il peso rappresenta una forza che si misura in newton (simbolo: N), mentre la massa si misura in kilogrammi (simbolo: kg).[4] Pertanto, quando si vuole rappresentare qualcosa che si esprime in multipli o sottomultipli del kilogrammo,[5] si fa riferimento a una massa.

## 15.3 Forme espressive particolari

- ridirezione

  È una questione di gusto personale, dal momento che molti preferiscono «re-direzione».[6]

- emettere attraverso lo standard output, emettere attraverso lo standard error

  Questa forma è quella usata nel documento. I motivi per cui è stata scelta sono tanti, ma non derivano da un'esperienza Unix. In generale, viene contestato che standard output e standard error sono file come gli altri, secondo la filosofia Unix, per cui su questi ci si «scrive».

## 15.4 Annotazioni per un uso futuro

Quelle che seguono sono annotazioni per un possibile uso futuro. Sono qui per non essere dimenticate.

- shadow password ---> ? (per ora solo password shadow)

- produttività

  Questo termine potrebbe essere utilizzato al posto di «velocità», quando si fa riferimento alla quantità di dati che possono transitare nell'unità di tempo. In altri termini, invece di parlare di velocità di un modem, si potrebbe parlare di produttività.

---

[4] $1\ N = 1\ kg*m/s^2$

[5] $1\ g = 10^{-3}\ kg$

[6] Il termine «ridirezione» viene usato anche in *IPv6* di Silvano Gai, McGraw Hill, 1997, alla sezione 6.4.3, anche se in questo caso si tratta di ridirezione dei pacchetti IPv6.

## 15.5 Nomi dei caratteri speciali

La tabella 15.6 elenca alcuni caratteri e simboli speciali, assieme alla denominazione usata in questo documento.

Tabella 15.6. Elenco dei nomi di alcuni caratteri e altri simboli.

| Simbolo | Denominazione |
|---|---|
| - | trattino (normale) |
| – | trattino basso |
| \| | barra verticale |
| / | barra obliqua (normale) |
| \ | barra obliqua (inversa) |
| ’ | apice singolo |
| ‘ | apice inverso |
| " | apice doppio, virgolette, virgolette alte |
| «, » | virgolette basse, virgolette uncinate |
| & | e-commerciale |
| ~ | tilde |
| @ | *at*, chiocciola, chiocciolina, chioccioletta -- meglio non usarlo |
| # | cancelletto -- meglio non usarlo |
| : | due punti (verticali) |
| .. | due punti in orizzontale |

In particolare, i simboli elencati di seguito meritano maggiore attenzione.

- @

  In origine questo simbolo è nato per abbreviare la parola latina «ad», mentre oggi si conosce prevalentemente la sua traduzione inglese: *at*. Sembra ricorrente il nome «chiocciola» in italiano, ma in generale non è il caso di nominarla in un testo scritto.

- #

  È difficile dare un nome a questo simbolo; attualmente è diffuso il termine «cancelletto» nel settore della telefonia, mentre è noto l’uso che se ne fa nell’ambito musicale, a rappresentare un diesis.

## 15.6 Riferimenti


- *Dictionnaire panlatin de l’informatique*

  *<http://www.tele3.net/dicoinfo/_bdt.htm>*

- *NetGlos - The Multilingual Glossary of Internet Terminology*

  *<http://wwli.com/translation/netglos/netglos.html>*

- *Amiga Translators’ Organization*

  *<http://bilbo.di.unipi.it/~ato-it/>*

- Silvano Gai, *IPv6*, McGraw-Hill, 1997

- Bureau International des Poids et Mesures, *Le Système international d’unités (SI)*

  *<http://www.bipm.org/pdf/brochure-si.pdf>*

- National Institute of Standards and Technology, *International System of Units (SI)*
  <*http://physics.nist.gov/cuu/Units/index.html*>
- National Institute of Standards and Technology, *Guide for the Use of the International System of Units (SI)*, 1995
  <*http://physics.nist.gov/cuu/pdf/sp811.pdf*>
- Markus Kuhn, *Standardized Units for Use in Information Technology*, 1995
  <*http://www.cl.cam.ac.uk/~mgk25/information-units.txt*>
- National Institute of Standards and Technology, *Prefixes for binary multiples*
  <*http://physics.nist.gov/cuu/Units/binary.html*>
- *Scienza, tecnologia e arte della stampa e della comunicazione*, Arti poligrafiche europee
  <*http://www.apenet.it/grafica/libri/Grafica/Grafica01/indice02.html*>
  <*http://www.apenet.it/grafica/libri/Grafica/Grafica02/indice02.html*>
  <*http://www.apenet.it/grafica/libri/Grafica/Grafica03/indice02.html*>
- *Scienza, tecnologia e arte della stampa e della comunicazione*: Giuseppe Pellitteri, Luigi Farinelli, *Grafismi*, Arti poligrafiche europee
  <*http://www.apenet.it/grafica/libri/Grafica/Grafica01/1123.html*>


## 15.7 Indice del glossario stilistico

numero di dispositivo, 186
numero primario, 186
numero secondario, 186
nvram, 185
octet, 189
offset, 201
on-line help, 195
opzione, 203
ora estiva, 182
orientamento, 200
orizzontale, 200
ottetto, 189
pacchetto, 188
package, 188
packet driver, 198
pagina di manuale, 195
parametro, 191
parametro formale, 191
parola chiave, 203
parola d’ordine, 203, 203
parole chiave, 203
parte frontale, 184
parte terminale, 184
partizione principale, 193
passphrase, 203
password, 203
patch, 188
path, 193
pathname, 193
pendenza, 199
percorso, 193
percorso assoluto, 193
percorso relativo, 193
permessi di accesso, 192
peso, 204
ping, 180
pipe, 180, 183
pipeline, 180, 183
pixel, 180
polizza, 200
porting, 192
portrait, 200
prefisso di formato, 198
primario, 198
principale, 198
print job, 184
privilegi, 194
procedura di accesso, 203
procedura di arresto del sistema, 184
procedura di avvio del sistema, 184
procedura di inizializzazione del sistema, 184

# Appendici

*Appendice* A

# Annotazioni su alcune sezioni particolari dell'opera

Si annotano qui delle informazioni sull'origine o lo stato di alcune sezioni dell'opera.

*Appendix* B

# GNU GENERAL PUBLIC LICENSE

non modificabile

Testo originale: <*http://www.fsf.org/copyleft/gpl.html*>

---

**GNU GENERAL PUBLIC LICENSE - Version 2, June 1991**



**Preamble**

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

**GNU GENERAL PUBLIC LICENSE - TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION**

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

> a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
>
> b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
>
> c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

> a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
>
> b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
>
> c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

**NO WARRANTY**

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PAR-

TICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

**Appendix: How to Apply These Terms to Your New Programs**

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) 19yy  <name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

# Indice analitico